O‘ZBEKISTON RESPUBLIKASI OLIY
VA O‘RTA MAXSUS TA’LIM VAZIRLIGI
O‘RTA MAXSUS, KASB-HUNAR TA’LIMI MARKAZI

Sh. Nazirov, A. Ne’matov, R. Qobulov

MA’LUMOTLAR BAZASINI DASTURLASH CHUQURLASHTIRILGAN KURSI

Axborot-kommunikatsiya texnologiyalari sohasidagi kasb-hunar kollejlarining «Axborot-kommunikatsiya tizimlari (3521916)» mutaxassisligi o‘quvchilari uchun o‘quv qo‘llanma

«SHARQ» NASHRIYOT-MATBAA
AKSIYADORLIK KOMPANIYASI
BOSH TAHRIRIYATI
TOSHKENT — 2007

N16

Mazkur o‘quv qo‘llanma Germaniya texnikaviy hamkorlik tashkiloti (GTZ) hamda Germaniya taraqqiyot banki (KfW) ishtirokidagi «Axborot-kommunikatsiya texnologiyalari sohasida kasb-hunar ta’limini rivojlantirishga ko‘maklashish» loyihasi doirasida ishlab chiqilgan.

O‘zbekiston Respublikasi Oliy va o‘rta maxsus ta’lim vazirligi O‘rta maxsus, kasb-hunar ta’limi markazi tomonidan axborot-kommunikatsiya texnologiyalari sohasidagi kasb-hunar kollejlari uchun tavsiya etilgan.

M u a l l i f l a r:

Sh. Nazirov — f.m.f.d., professor
A. Ne’matov — f.m.f.n., TATU dotsenti
R. Qobulov — f.m.f.n., TATU dotsenti

M a s’ u l m u h a r r i r

Sh. A. Nazirov
fizika-matematika fanlari doktori, professor

T a q r i z c h i l a r:

M. Aripov
O’zMU professori, fizika-matematika fanlari doktori

M. E. Zaynutdinova
Mirzo Ulug‘bek nomidagi informatika
kasb-hunar kolleji «Informatika va dasturlash»
kafedrasi mudiri, maxsus fan o‘qituvchisi

Nazirov Sh.

Ma’lumotlar bazasini dasturlash chuqurlashtirilgan kursi: Kasb-hunar kollejlarining «axborot-kommunikatsiya tizimlari (3521916)» mutaxassisligi o‘quvchilari uchun o‘quv qo‘l. / Sh. Nazirov, A. Ne’-matov, R. Qobulov; Mas’ul muharrir Sh. Nazirov; O‘zbekiston Respublikasi Oliy va o‘rta maxsus ta’lim vazirligi, O‘rta maxsus, kasb-hunar ta’limi markazi. — T.: Sharq, 2007. — 136 b.

I. Ne’matov A. II Qobulov R.

BBK 32.937-018ya722

ISBN 978-9943-00-235-7

MUNDARIJA

KIRISH

Ma'lumotlar bazasi dunyosi tobora yagona bo'lib bormoqda. Bu jarayon har xil kompyuter muhitlarida faoliyat ko'rsatuvchi axborot tizimlarini yaratishda qo'llanuvchi yagona standart til yaratishni talab qildi. Standart til bir komandalar to'plamini bilgan foydalanuvchilarga ularni shaxsiy kompyuter, tarmoq ishchi stantsiyasi yoki katta EHM da ishlashlaridan qat'iy nazar ma'lumotni yaratish, izlash va uzatishga imkon beradi.

Internetga asoslanuvchi ma'lumotlar bazalarini yaratish hozirgi davrda keng rivojlanib bormoqda. Ushbu qo'llanmada SQL tili asoslari, PHP tili hamda MySQL MBBT si haqida to'liq ma'lumotlar berilgan. Qo'llanma to'rt qismdan iborat bo'lib, birinchi qismda SQL, PHP va MySQL xarakteristikalari va tarixi keltirilgan.

Ikkinchi qismda SQL tili asoslari, DDL va DML komandalari, mantiqiy operatorlar, bir necha jadvallar bilan ishlash, so'rovlarda guruhlash va funksiyalardan foydalanish ko'rib chiqilgan. Bundan tashqari ushbu qismda, foydalanuvchilar va ular imtiyozlari, tarmoqda ma'lumotlar bazalari arxitekturasi va ODBC interfeysi haqida ma'lumotlar keltirilgan.

Uchinchi qism PHP tiliga bag'ishlangan bo'lib, PHP tilining operatorlari, PHP da massivlar va funksiyalar hamda, fayllar bilan ishlash k o'rib chiqilgan. Amaliy misollarda dinamik WEB sahifalar yaratish ko'rsatilgan.

To'rtinchi qismda MySQL serveri bilan ishlash, murojaat shartlarini tekshirish va PHP boshqariluvchi so'rov yaratish haqida ma'lumotlar keltirilgan.

Ushbu o'quv qo'llanma kasb-hunar kollejlari o'qituvchilari va o'quvchilari uchun mo'ljallangan bo'lib, shu bilan birga oliy o'quv yurtlari professor o'qituvchilari va talabalari hamda hamma shu sohaga qiziquvchilar tomonidan foydalanilishi mumkin.

SQL

SQL (Structured Query Language, odatda «sikvel» deyiladi) ma'nosi *Tarkiblangan so'rovlar tili*. Bu relyatsion ma'lumotlar bazalarida ishlashga imkon beradigan tildir. Bu til ifodalarining xususiyati shundan iboratki, ular ma'lumotlarni qayta ishlash protseduralariga emas, natijalariga yo'naltirilgandir. SQL o'zi ma'lumotlar qayerda joylashgani, qanday indekslar va hatto amallarning eng effektiv ketma-ketligini qo'llash kerakligini aniqlaydi; bu detallarni ma'lumotlar bazasiga so'rovlarda ko'rsatish kerak emas.

SQL tilining o'zi IBM kompaniyasida MBBT DB2 yaratish jarayonida ishlab chiqilgan va keng ko'lamda RISC protsessorli mashinalarda UNIX tizimlar asosida, hamda meynfreymlarda, superkompyuterlar asosida qurilgan katta hisoblash tizimlarida qo'llanilgan.

Shu bilan birga mustaqil bo'lmasdan PL/SQL, va Transact-SQL kabi ichki dasturlash tillariga inkapsulyatsiya qilinadi. 1986-yilda, ANSI (American National Standart Institute) SQL tilining rasmiy standartini ishlab chiqdi, 1992-yil bu standart kengaytirildi. Butun til 30 ga yaqin operatorlarga ega bo'lib, ba'zi versiyalarida sal ko'proq, ba'zilarida sal kamroq. Har qanday MB har xil obyektlarga ega, Ya'ni jadvallar, protseduralar, funksiyalar, tasavvurlar, ketma-ketliklar va hokazo.

«Klient-Server» texnologiyasiga ko'ra, foydalanuvchi EHM (Klient) lar so'rovlari maxsus ma'lumotlar serverlarida (Server) qayta ishlanadi, foydalanuvchi EHM larga faqat so'rovni qayta ishlash natijalari qaytariladi.

Tabiiyki Server bilan muloqot qilish uchun yagona til kerak va bunday til sifatida SQL tanlandi. Shuning uchun hamma zamonaviy relyatsion MBBT versiyalari (DB2, Oracle, Ingres, Informix, Sybase, Progress, Rdb) va hattoki norelyatsion MBBT versiyalari (masalan, Adabas) «Klient-Server» texnologiyasi va SQL tilidan foydalanadilar.

SQL tilida ma'lumotlarni jadval ko'rinishda tasvirlashga yo'naltirilgan amallar kontsepsiyasi ko'p bo'lmagan (30 dan kam) ifodalardan iborat kompakt til yaratishga imkon berdi.

Ikki xil SQL mavjud: **Interaktiv** va **Joylashtirilgan.** Ko‘p hollarda ikkala forma bir xil ishlaydi, lekin ikki xil foydalaniladi:

***Interaktiv* SQL** ma’lumotlar bazasi o‘zida faoliyat ko‘rsatadi va buyurtmachi foydalanishi uchun chiqish hosil qilish uchun ishlatiladi. SQL bu formasida, siz komanda kiritsangiz, u darov bajariladi va siz darhol natijani(agar u mavjud bo‘lsa) ko‘rishingiz mumkin.

***Joylashtirilgan* SQL** boshqa tilda yaratilgan dasturga joylashtirilgan SQL komandalardan iborat.

SQL interaktiv va joylashtirilgan formalarida ko‘p sonli guruhlar yoki subbo‘limlar mavjud. Ular ANSI tomonidan e’tiborga olingan va konseptual darajada foydali, lekin ko‘pchilik SQL dasturlar ularni alohida qayta ishlamaydi, shuning uchun ular aslida SQL komandalarining funksional kategoriyalaridir.

- **DDL** *(Ma’lumotlarni Ta’riflash Tili)* — ANSI da sxemani ta’riflash tili, obyektlarni (jadvallar, indekslar, tasavvurlar va hokazo) yaratuvchi komandalardan iborat.
- **DML** (*Ma’lumotlarni O‘zgartirish Tili*) — bu ixtiyoriy daqiqada jadvallarda qanday qiymatlar saqlanishini aniqlovchi komandalar majmuasidir.
- **DCD** (*Ma’lumotlarni Boshqarish Tili*) foydalanuvchiga ma’lum obyektlar ustida ma’lum ta’sir o‘tkazishga ruxsat berish yoki bermaslikni aniqlovchi vositalardan iborat.
- **SQL** Standarti **ANSI** tomonidan aniqlangan va hozirda **ISO** tomonidan qabul qilingan. Lekin kommersial ma’lumotlar bazalari dasturlari ANSI ni ogohlantirmasdan SQL ni kengaytiradilar, ya’ni foydali hisoblagan har xil xossalar qo‘shadilar.

PHP

PHP/FI.

PHP boshqa mahsulot, PHP/FI rivojlantirilishi natijasida yaratildi. PHP/FI 1995-yilda Rasmus Lerdorf tomonidan o‘zining online-rezyumesiga murojaatni kuzatish uchun Perl-scriptlar sodda to‘plami sifatida yaratildi.

U bu scriptlar to‘plamini ‘Personal Contents Page Tools’ deb atadi. Katta funksionallik talab qilingani uchun, Rasmus ma’lumotlar bazasi bilan ishlash imkoniga ega bo‘lgan C kengaytirilgan realizatsiyasini yaratdi va foydalanuvchilarga

sodda dinamik Web-ilovalar yaratish imkonini berdi. Rasmus har bir foydalanuvchi kengaytirish va o‘zgartirish imkoniyatiga ega bo‘lishi uchun PHP/FI kodini keng ommaga e’lon qildi.

PHP/FI, Personal Contents Page / Forms Interpreter, hozirgi PHP asosiy funksionalligiga ega edi. U Perl kabi o‘zgaruvchilar, forma o‘zgaruvchilari avtomatik interpretatsiyasi va HTML ga qo‘llangan sintaksisga ega edi. Sintaksis o‘zi Perl ni eslatardi, faqat cheklangan, soddalashtirilgan va to‘la bo‘lmagan.

1997-yilda PHP/FI 2.0, C-realizatsiya ikkinchi versiyasi, butun dunyo bo‘yicha bir necha ming muxlislarga ega bo‘lib, taxminan 50,000 domenlarga o‘rnatilgan edi. Bu hamma Internet domenlarning 1% ini tashkil qilar edi. Ko‘p odamlar o‘zlarining kod bloklarini bu loyiha uchun taklif qilganlari bois u bir kishining loyihasi bo‘lmay qoldi.

PHP/FI 2.0 rasmiy ravishda faqat 1997-yil noyabrida chiqarildi. Ungacha u asosan beta-relizlar shaklida mavjud edi. Shundan so‘ng ko‘p o‘tmasdan birinchi alpha PHP 3.0 paydo bo‘ldi.

PHP 3.

PHP 3.0 bugungi PHP ga o‘xshagan birinchi versiya edi. Uni Andi Gutmans va Zeev Suraski 1997-yilda to‘la qaytadan yozilgan til sifatida yaratdilar, chunki ular PHP/FI 2.0 tilini o‘zlarining eCommerce-ilovalarini yaratish uchun yetarli imkoniyatga ega emas deb topdilar. Kuchlarni birlashtirish uchun, Andi, Rasmus va Zeev PHP 3.0 ni PHP/FI 2.0 rasmiy vorisi sifatida yaratdilar va e’lon qildilar. Natijada PHP/FI 2.0 rivojlanishi to‘xtadi.

PHP 3.0 eng kuchli tomonlaridan biri uni kengaytirish imkoni edi. Chekli foydalanuvchilarga har xil ma’lumotlar bazalari, protokol va API lar uchun mustahkam infrastruktura, hamda kengaytirish imkoniyatini yaratgani uchun, o‘nlab foydalanuvchilarni yangi kengaygan modullar yaratishga undar edi. Balkim PHP 3.0 o‘ta mashhurligi sababi shundadir. PHP 3.0 asosiy xusuiyatlaridan biri obyektga yo‘naltirilgan sintaksis edi.

Tilga PHP: Hypertext Preprocessor deb nom berildi.

1998-yil oxiriga kelib PHP o‘n minglab foydalanuvchilar va yuz minglab Web-saytlar uchun asos bo‘lib qoldi. Eng mashhur bo‘lgan paytda PHP 3.0 taxminan Internet Web-serverlarining 10% iga o‘rnatilgan edi .

PHP 3.0 rasmiy ravishda 1998-iyunida 9 oylik oshkora testlashdan so‘ng chiqarilgan edi.

PHP 4.

1998-yil qishida PHP 3.0 rasmiy e’lon qilingandan so‘ng, Andi Gutmans va Zeev Suraski katta amaliy dasturlar bilan ishlashda unumdorligini oshirish va PHP kodli bazasi modulligini oshirish maqsadida PHP yadrosini qayta ishlashga kirishdilar. Bunday Amaliy dasturlar yaratish PHP 3.0 da mumkin edi, lekin PHP 3.0 murakkab kompleksli amaliy dasturlarga xizmat qilish uchun yaratilmagan edi.

Yangi mashina, «Zend Engine» (yaratuvchilar nomlari asosida — Zeev va Andi), bu masalalarni muvaffaqiyatli hal qildi va 1999-o‘rtasida paydo bo‘ldi. Bu mashinaga asoslangan va ko‘pchilik yangi imkoniyatlarga ega bo‘lgan PHP 4.0, rasmiy ravishda 2000-yil mayida, PHP 3.0 dan ikki yil keyin chiqarildi.

Ancha oshgan unumdorlikdan tashqari bu versiyada PHP 4.0 quyidagi imkoniyatlarni kiritdi: katta sondagi Web-serverlar, HTTP-sessiyalarni qo‘llash, chiqarishni buferlash, foydalanuvchi kiritishi bilan xavfsiz ishlash usullari va tilning turli yangi konstruksiyalari.

PHP 4 hozirda PHPning oxirgi versiyasidir. Zend Engineni PHP 5.0 ga integratsiya qilish uchun yaxshilash va modifikatsiya qilish ustida ish boshlangan.

Hozirgi kunda PHP yuz minglab dasturchilar tomonidan foydalaniladi va bir necha million saytlar uning o‘rnatilgani haqida ma’lumot beradilar, bu esa Internet domenlarning 20% idan ortiqdir.

PHP yaratuvchilari komandasi o‘nlab dasturchilardan hamda PHP bilan bog‘liq PEAR va xujjatlash loyihalari kabi loyihalar ustida ishlovchilardan iborat.

MySQL

Ma’lumotlar bazasi va WWW.

Ma’lumotlar bazasi o‘ta tez rivojlangan hamda MySQL va mSQL o‘zini ko‘rsatgan soha, Internet uchun dasturlar yaratishdir. Internet uchun murakkab va ishonchli dasturlarga ehtiyoj oshgan sari ma’lumotlar bazasiga ehtiyoj ham oshib bormoqda. Server ma’lumotlar bazasi Internetda ko‘p funktsiyalarni qo‘llashi mumkin. Har qanday veb- sahifa ma’lumotlar bazasi tomonidan boshqarilishi mumkin.

Misol tariqasida o‘z katalogini WWW da e’lon qilmoqchi bo‘lgan va Internet orqali buyurtmalar qabul qilmoqchi bo‘lgan katalog bo‘yicha sotuvchini ko‘raylik.

Agar katalogni HTML-fayllar shaklida e’lon qilinsa yangi tovar qo‘shilganda yoki narx o‘zgarganda kimdir katalogni tahrirlashi lozim bo‘ladi. Agar buning o‘rniga katalog ma’lumotlarini relyatsion ma’lumotlar bazasida saqlansa katalogdagi o‘zgarishlarni ma’lumotlar bazasidagi tovar yoki narx haqidagi ma’lumotlarni o‘zgartirish yo‘li bilan real vaqt masshtabida e’lon qilish imkoniyati tug‘iladi.

Bundan tashqari katalogni mavjud buyurtmalarni qayta ishlash elektron tizimlari bilan integratsiya qilish imkoniyati tug‘iladi. Shunday qilib bunday veb-saytni boshqarish uchun ma’lumotlar bazasidan foydalanish sotuvchiga ham, oluvchiga ham qulayliklar tug‘diradi.

Shu tarzda veb-sahifa ma’lumotlar bazasi bilan bog‘lanadi. Ma’lumotlar bazasi sizni veb-serveringizda yoki sizni serveringiz ma’lumot almashishi mumkin bo‘lgan boshqa mashinada joylashgan bo‘lishi mumkin. (Yaxshi MBBT bunday vazifalarni taqsimlashni oson tashkil qila oladi). Siz o‘zingizning veb-sahifangizga forma joylashtirasiz va foydalanuvchi uzatish kerak bo‘lgan so‘rov yoki ma’lumotni shu formaga kiritadi. Formani serverga uzatgandan so‘ng, server siz yozgan dasturni ishga tushiradi va bu dastur foydalanuvchi uzatgan ma’lumotlarni ajratib oladi. Bu dasturlar ko‘pincha CGI-ssenariylar yoki Java da server dasturlari shaklida yaratiladi, lekin dasturni HTML-sahifaga to‘g‘ridan to‘g‘ri joylashtirish ham mumkin.

Endi sizning dasturingiz foydalanuvchiga qanday ma’lumotlar kerak va u ma’lumotlar bazasiga nima kiritmoqchiligini biladi. Dastur ma’lumotlarni tanlash yoki o‘zgartirish uchun SQL komanda yaratadi, ma’lumotlar bazasi bo‘lsa qolganini bajaradi. Ma’lumotlar bazasidan olingan natijalarni sizning dasturingiz yangi HTML-sahifa shakliga keltirib qaytadan foydalanuvchiga yuboradi.

mSQL tarixi.

To 1994-yilgacha SQL qo‘llovchi RMBBT sotib olish uchun ancha ko‘p pul ketkazishga to‘g‘ri kelar edi. Bozorda Oracle, Sybase va Informix hokimlik qilar edi.Bu ma’lumotlar bazasini boshqarish tizimlari murakkab bog‘lanishlarga ega bo‘lgan katta hajmdagi ma’lumotlarni qayta ishlash uchun mo‘ljallangan edi.

Ular katta quvvatga va imkoniyatlarga ega bo‘lib, katta hisoblash resurslarini talab qilar edilar va narxi qimmat edi. U paytda 2000$ ga 200-MHz Pentium li server sotib olish mumkin emas edi. Bunday MBBT uchun kerakli resurslar o‘ng minglab dollar turar edi.

Katta korporatsiyalar va yirik universitetlar uchun bunday serverlar komplektlari va MBBT lar uchun bir necha million dollar sarflash muammo tug‘dirmas edi.

Kichik tashkilotlar va xususiy foydalanuvchilar ojiz kichik amaliy dasturlardan foydalanishga majbur edilar. Bir nechta arzon klient/ server arxitekturali MBBT lar o‘sha paytda mavjud edi, lekin ularning hech biri so‘rovlar tili sifatida SQL dan foydalanmas edi. Eng ko‘zga ko‘ringanlaridan biri Ingres kommertsial ma’lumotlar bazasi bilan bitta ajdodga ega bo‘lgan Postgres edi. Lekin baxtga qarshi kommertsial analoglari kabi resurslarni talab qilardi va SQL dan so‘rovlar tili sifatida foydalanish imkoniyatini bermas edi. O‘sha paytda Postgresda QUEL tilining ko‘rinishi bo‘lgan PostQUEL tilidan foydalanardi.

Devid Xyuz.

Devid Xyuz (David Hughes) (yana Bamby sifatida ma’lum) Avstraliyada Bond Universitetida yozgan dissertatsiyasining bir qismi monitoring tizimini yaratish va tizimlar guruhini bir yoki bir necha joydan boshqarishga bag‘ishlangan edi. Loyiha Minerva Network Management System deb nomlangan edi. Minerva asosiy elementi tarmoqdagi hamma kompyuterlar haqidagi ma’lumotlarni saqlovchi ma’lumotlar bazasi edi. Universitet talabasi bo‘lgani va katta kommertsial ma’lumotlar bazalari ishlaydigan serverlarga murojaat qilish imkoniyati yo‘q bo‘lgani uchun, Xyuz uning talablariga Postgres — javob beradi degan qarorga keldi.

Uning hamkasblari SQL tilidan Minerva uchun standart so‘rovlar tili sifatida foydalanishni taklif qilishdi. SQL ga asoslangan holda Minerva dunyoning SQL ni qo‘llovchi MBBT mavjud ixtiyoriy nuqtasida qo‘llanishi mumkin edi. Boshqacha qilib aytganda Postgres foydalanuvchilari bilan chegaralab qo‘ygan PostQUEL ga nisbatan SQL Minerva uchun kengroq foydalanuvchilar bilan ishlashga imkon berar edi. Oxiri kelib hatto Postgres ham bugun SQL ni qo‘llaydi.

Bir tomondan SQL standartidan foydalanish istagi va

boshqa tomondan SQLni qo‘llovchi ma’lumotlar bazasiga murojaat qilish imkoniyati yo‘qligi, Xyuzni qiyin ahvolga solib qo‘ydi. Agar Minervada SQLga asoslangan so‘rovlar tilidan foydalanilsa, mos ishlash mexanizmiga ega MBBT topib bo‘lmaydi. Qimmat RMBBT sotib olish imkoniyatiga ega bo‘lmagan Xyuz masalaning ajoyib echimini topdi: SQL so‘rovlarni PostQUEL so‘rovlariga translyatsiya qiluvchi dastur yaratish lozim edi. Bu dastur Minervaga uzatilgan SQL so‘rovlarni ilib olishi, PostQUELga aylantirishi va natijani Postgresga uzatishi kerak edi. Xyuz shunday dastur yaratdi va uni miniSQL yoki mSQL deb atadi.

PostQUEL translyatoridan RMBBT ga.

Bir necha davr mobaynida bu konfiguratsiya Xyuzni qanoatlantirar edi. Minerva uchun agar SQLni tushunsa qanday MBBT dan foydalanishning farqi yo‘q edi va u Postgres SQLni tushunadi deb hisoblar edi, chunki o‘rtada PostQUELga translyatsiya qiluvchi mSQL joylashgan edi. Baxtga qarshi Minerva o‘sishi bilan uning ishi qiyinlashib bordi. Aniq bo‘ldiki, na Postgres, na boshqa katta RMBBT Minerva uchun kerak bo‘lgan chekli resurslar asosida kam sonli imkoniyatlarni qo‘llay olmas edi. Masalan, Minerva uchun bir vaqtning o‘zida bir necha ma’lumotlar bazasiga ulanish talab qilinar edi. Buning uchun Postgres bir vaqtning o‘zida ma’lumotlar bazasi serveri bir necha nusxasini ishga tushirishni talab qilar edi. Bundan tashqari bir necha potensial loyiha qatnashchilari unda qatnasha olmas edilar, chunki Postgres ularning tizimlarini qo‘llamas edi, ular bo‘lsa SQLga asoslangan qimmat RMBBT sotib olishga imkonlari yo‘q edi.

Bu muammolarga duch kelgandan so‘ng Postgresga munosabatini o‘zgartirdi. O‘zining kattaligi va murakkabligi bilan Minerva talablaridan yuqori edi. Minerva tomonidan generatsiya qilinadigan so‘rovlar asosan INSERT, DELETE va SELECT sodda operatorlaridan iborat edi. Postgres da mavjud va unumdorlikni kamaytiruvchi qolgan hamma imkoniyatlar Minerva uchun kerak emas edi.

Xyuzda SQLga translyatsiyani amalga oshiruvchi mSQL mavjud edi. Unga talablariga javob beruvchi ma’lumotlar bazasi serverini yaratish uchun ma’lumotlar ombori va ma’lumotlarni ajratib olish imkoniyatini qo‘shish qolgan edi. Bu evolyutsiya bugungi kunda mavjud mSQL ga olib keldi.

MySQL tarixi.

MySQL ni faqat mSQL kamchiliklariga javob sifatida qarash noto‘g‘ridir. Uning ixtirochisi Maykl Videnius (yana Monty sifatida ma’lum) shved kompaniyasi TsX xodimi, ma’lumotlar bazasi bilan 1979-yildan beri ishlaydi. Yaqin paytgacha Videnius TsX da faqat dasturchi edi. 1979-yilda firma ichida foydalanish uchun UNIREG nomli ma’lumotlar bazasini boshqarish vositasini yaratdi. 1979-yildan so‘ng UNIREG bir necha tillarda yozildi va katta ma’lumotlar bazalarini qo‘llash uchun kengaytirildi.

Bitta dastur bajarilayotgan har bir jarayon bu dastur nusxasi deyiladi, chunki xuddi o‘zgaruvchi nusxasi kabi xotiradan joy oladi.

1994-yilda TsX WWW uchun Amaliy dasturlar yarata boshladi va bu loyihani qo‘llashda UNIREG dan foydalandi. Baxtga qarshi, UNIREG katta harajatlar talab qilgani uchun, undan veb-sahifalarni dinamik generatsiya qilish uchun muvaffaqiyatli foydalanib bo‘lmadi. TsX shundan so‘ng SQL va mSQL ga murojaat qildi. Lekin o‘sha paytda mSQL faqat 1.x relizlari shaklida mavjud edi. Yuqorida aytganimizdek mSQL 1.x versiyalari hech qanday indekslarni qo‘llamas edi va shuning uchun UNIREG dan unumdorligi past edi.

Videnius mSQL muallifi Xyuz bilan bog‘landi va mSQL ni UNIREG dagi V+ ISAM qayta ishlovchisiga ulash fikri bilan qiziqtirmoqchi bo‘ldi. Lekin Xyuz shu paytga kelib mSQL2 yaratish yo‘lida ancha ilgari ketgan va indekslar bilan ishlash vositalarini yaratgan edi. TsX o‘z talablariga ko‘proq mos keluvchi ma’lumotlar bazalari serverini yaratishga qaror qildi.

TsX xodimlari yangidan velosiped ixtiro qilib o‘tirmadilar. Ular UNIREG ni asos qilib oldilar va soni oshib borayotgan o‘zga dasturchilar utilitalaridan foydalandilar. O‘z tizimlari uchun boshida mSQL uchun yaratilgan API bilan deyarli ustmaust tushuvchi API yaratdilar. Natijada yangi, kengroq imkoniyatga ega TsX ma’lumotlar bazasiga o‘tmoqchi bo‘lgan mSQL foydalanuvchisi o‘z kodiga juda kam o‘zgartirish kiritishi talab qilinardi. Shu bilan birga Yangi ma’lumotlar bazasi kodi to‘la original edi.

1995-yil may oyiga kelib TsX kompaniya ichki talablarini qanoatlantiruvchi ma’lumotlar bazasi — MySQL 1.0 ga ega edi. Firma biznes-partneri Detron HB dan David Aksmark (David Axmark) TsX ga o‘z serverini Internetda ko‘rsatishni taklif qildi.

Serverni Internetda ko‘rsatishdan maqsad — birinchi bo‘lib Aladdin Peter Deych (Aladdin Peter Deutsch) qo‘llagan biznes modeldan foydalanishdir. Natijada MySQLni mSQL ga nisbatan «tekinroq» qiluvchi o‘ta moslashuvchan mualliflik huquqlari olindi.

Nomiga kelganda Videnius bu haqida shunday deydi: «MySQL nomi qayerdan kelib chiqqani hozirgacha noma’lum. TsX da asosiy katalog hamda bibliotekalar va utilitalar ko‘p qismi bir necha o‘n yildan beri «mu» prefiksiga ega. Shu bilan birga mening qizimning ismi ham May (My). Shuning uchun bu ikki manbaning qaysi biri MySQL nomini berganligi haligacha sir».

MySQL ni Internetda e’lon qilingandan beri u ko‘pgina UNIX-tizimlarga, Win32i OS/2 ga ko‘chirildi. TsX kompaniyasi fikricha, MySQL ni 500 000 ga yaqin serverlar ishlatadi.

2.1. SQL TILIDA JADVALLAR BILAN ISHLASH

SQL tilida ma'lumotlar turlari.

SQL tilida quyidagi asosiy ma'lumotlar turlari ishlatilib, ularning formatlari har xil MBBT lar uchun farq qilishi mumkin:

INTEGER	— butun son (odatda 10 tagacha qiymatli raqam va ishora).
SMALLINT	— «qisqa butun» (odatda 5 tagacha qiymatli raqam va ishora).
DECIMAL(p,q)	— o'nli son, p raqam va ishoradan iborat ($0 < p < 16$). O'nli nuqtadan so'ng raqamlar soni q orqali beriladi ($q < p$, agar $q = 0$ bo'lsa, tashlab yuborilishi mumkin).
FLOAT	— haqiqiy son 15 ta qiymatli raqam va butun darajadan iborat. Daraja MBBT tipi bilan aniqlanadi (masalan, 75 yoki 307).
CHAR(n)	— uzunligi o'zgarmas, n ga teng bo'lgan simvolli qator ($0 < n < 256$).
VARCHAR(n)	— uzunligi o'zgaruvchi, n simvoldan oshmagan simvolli qator ($n > 0$ va har xil MBBT larda har xil, lekin 4096 dan kam emas).
DATE	— maxsus komanda orqali aniqlanuvchi formatdagi sana; sana maydonlari bizning eramizdan oldin bir necha mingyilliklardan boshlanuvchi va bizning eramiz beshinchi-o'ninchi mingyilligi bilan cheklangan haqiqiy sanalarni o'z ichiga olishi mumkin.
TIME	— maxsus komanda orqali aniqlanuvchi formatdagi vaqt (ko'zda tutilgan bo'yicha hh.mm.ss).
DATETIME	— sana va vaqt kombinatsiyasi.
MONEY	— maxsus komanda orqali aniqlanuvchi formatdagi pul.

Jadvallarni yaratish.

Jadvallar CREATE TABLE komandasi bilan yaratiladi. Bu komanda qatorlarsiz bo‘sh jadval yaratadi. CREATE TABLE komandasi jadval nomini va jadval o‘zini ma’lum tartibda ko‘rsatilgan ustunlar nomlari ketma-ketligi ta’rifi ko‘rinishida aniqlaydi. U ma’lumotlar tiplari va ustunlar o‘lchovini aniqlaydi. Har bir jadval juda bo‘lmaganda bitta ustunga ega bo‘lishi kerak.

CREATE TABLE komandasi sintaksisi:

CREATE TABLE <table-name >
(<column name> <data type>[(<size>)],
<column name> <data type>[(<size>)], ...)

Argument qiymati kattaligi ma’lumot turiga bog‘liqdir. Agar siz maxsus ko‘rsatmasangiz, tizim avtomatik qiymatni o‘rnatadi.

Bundan buyon quyida keltirilgan 3 ta jadvaldan iborat ma’lumotlar bazasini ko‘ramiz.

Sotuvchilar jadvali (Salepeople):

Snum	Sname	City	Comm
11	Peel	London	0.12
12	Serres	San Jose	0.13
14	Motika	London	0.11

SNum — har bir sotuvchi unikal nomeri,

SName — sotuvchi nomi,

City — sotuvchi adresi (shahar),

Comm — sotuvchilarning o‘nli shakldagi komission foydasi.

Buyurtmachilar jadvali (Customers):

Cnum	Cname	City	Rating	SNum
21	Hoffman	London	100	11
22	Giovanni	Rome	200	13
23	LiuSan	Jose	200	12

CNum — har bir buyurtmachi unikal nomeri,

CName — buyurtmachi nomi,

City — buyurtmachi adresi (shahar),

Rating — buyurtmachining boshqalardan ustunlik darajasini ko‘rsatuvchi kod (reyting),

SNum — shu buyurtmachiga tayinlangan sotuvchi nomeri.

Buyurtma jadvali (Orders):

ONum	AMT	ODate	CNum	SNum
38	4723.00	1990/10/05	26	11
310	1309.95	1990/10/06	24	12

ONum — har bir sotib olish unikal nomeri,

AMT — sotib olish summasi qiymati,

ODate — sotib olish sanasi,

CNum — sotib oluvchi buyurtmachi nomeri,

SNum — sotuvchining nomeri.

Misol uchun sotuvchilar jadvalini yaratishni ko‘rib chiqamiz:

```
CREATE TABLE Salepeople
(SNum    integer,
SName    char (10),
City     char (10),
Comm     decimal)
```

Jadvallarni o‘chirish.

Jadvalni o‘chirish imkoniga ega bo‘lish uchun, jadval egasi (ya’ni yaratuvchisi) bo‘lishingiz kerak. Faqat bo‘sh jadvalni o‘chirish mumkin. Qatorlarga ega bo‘lgan, to‘ldirilgan jadvalni o‘chirish mumkin emas, Ya’ni jadval o‘chirishdan oldin tozalangan bo‘lishi kerak. Jadvalni o‘chirish komandasi quyidagi ko‘rinishga ega:

DROP TABLE < table name >;

Masalan: **DROP TABLE Salepeople**

Jadvalni yaratilgandan so‘ng o‘zgartirish.

Jadvalni o‘zgartirish uchun ALTER TABLE komandasidan foydalaniladi. Bu komanda jadvalga yangi ustunlar qo‘shish, ustunlarni o‘chirish, ustunlar kattaligini o‘zgartirish hamda cheklanishlarni qo‘shish va olib tashlash imkoniyatlariga ega. Bu komanda ANSI standarti qismi emas, shuning uchun har xil tizimlarda har xil imkoniyatlarga ega.

Jadvalga ustun qo‘shish uchun komandaning tipik sintaksisi:

ALTER TABLE <table name> ADD <column name> <data type> <size>;
Masalan:
ALTER TABLE Salepeople ADD Phone CHAR(7)

2.2. JADVALLAR UCHUN CHEKLANISHLAR

Cheklanishlarni kiritish.

Jadval yaratayotganingizda (yoki uni o‘zgartirayotganingizda), siz maydonlarga kiritilayotgan qiymatlarga cheklanishlar o‘rnatishingiz mumkin. Bu holda SQL cheklanishlarga to‘g‘ri kelmaydigan hamma qiymatlarni rad etadi. Cheklanishlar ikki asosiy turi mavjud: — ustun va jadval cheklanishlari. Ularning farqi shundaki ustun cheklanishi faqat ayrim ustunlarga qo‘llanadi, jadval cheklanishi bo‘lsa bir yoki bir necha ustunlar guruhiga qo‘llanadi. Ustun cheklanishi ustun nomi oxiriga ma’lumotlar tipidan so‘ng va verguldan oldin qo‘yiladi. Jadval cheklanishi jadval nomi oxiriga so‘nggi dumaloq verguldan oldin qo‘yiladi. Cheklanishlar hisobga olingan CREATE TABLE komandasi sintaksisi:

CREATE TABLE < table name >
(<column name> <data type> <column constraint>,
<column name> <data type> <column constraint> ...
<table constraint> (<column name>
[, <column name>])...)

Maydonga bo‘sh (NULL) qiymatlar kiritilishining oldini olish uchun CREATE TABLE komandasida NOT NULL cheklanishi ishlatiladi. Bu cheklanish faqat har xil ustunlar uchun o‘rnatiladi.

Masalan, shu narsa aniqki, birlamchi kalitlar hech qachon bo‘sh bo‘lmasliklari kerak, shuning uchun Salepeople jadvalini quyidagicha yaratish mumkin:

CREATE TABLE Salepeople
(SNum integer NOT NULL,
SName char (10),
City char (10),
Comm decimal)

Ko‘p hollarda ustunga kiritilgan qiymatlar bir-biridan farq qilishi kerak. Agar ustun uchun UNIQUE cheklanishi o‘rnatilsa, bu ustunga mavjud qiymatni kiritishga urinish rad etiladi.

Bu cheklanish bo‘sh bo‘lmaydigan (NOT NULL) deb e’lon qilingan maydonlarga qo‘llanishi mumkin.

Masalan:

CREATE TABLE Salepeople
(SNum integer NOT NULL UNIQUE,
SName char (10),
City char (10),
Comm decimal)

Unikalligi talab qilinadigan maydonlar (birlamchi kalitlardan tashqari) kandidat kalitlar yoki unikal kalitlar deyiladi.

Jadval cheklanishi UNIQUE maydonlar guruhiga o‘rnatilishi mumkin. Bu bir necha maydonlar qiymatlari kombinatsiyasi unikalligini ta’minlaydi. Bizning ma’lumotlar bazamizda har bir buyurtmachi bitta sotuvchiga biriktirilgan. Ya’ni Buyurtmachilar jadvalida buyurtmachi nomeri (CNum) va sotuvchi nomeri (snum) kombinatsiyasi unikal bo‘lishi kerak. Bu cheklanishni UNIQUE (CNum, SNum) yordamida, Customers jadvalini yaratishda kiritish mumkin. Bu ustunlar uchun NOT NULL cheklanishini kiritish zarurdir.

Birlamchi kalitlar cheklanishlari.

SQL birlamchi kalitlarni to‘g‘ridan to‘g‘ri birlamchi kalit (PRIMARY KEY) cheklanishi orqali ta’riflaydi. PRIMARY KEY jadvalni yoki ustunlarni cheklashi mumkin. Bu cheklanish UNIQUE cheklanishi kabi ishlaydi, jadval uchun faqat bitta birlamchi kalit (ixtiyoriy sondagi ustunlar uchun) aniqlanishi mumkin bo‘lgan holdan tashqari. Birlamchi kalitlar NULL qiymatga ega bo‘lishi mumkin emas.

Misol:

CREATE TABLE Salepeople
(SNum integer NOT NULL PRIMARY KEY,
SName char (10),
City char (10),
Comm decimal)

PRIMARY KEY cheklanishi qiymatlar unikal kombinatsiyasini tashkil qiluvchi bir necha maydonlar uchun qo‘llanishi mumkin. Masalan PRIMARY KEY cheklanishini juftliklar uchun qo‘llash mumkin:

CREATE TABLE Namefield
(firstname char (10) NOT NULL,
lastname char (10) NOT NULL

city char (10),
PRIMARY KEY (firstname, lastname))

Maydon qiymatlarini tekshirish (CHECK cheklanishi).

CHECK cheklanishi jadvalga kiritilayotgan ma'lumot qabul qilinishidan oldin mos kelishi lozim bo'lgan shart kiritishga imkon beradi. CHECK cheklanishi CHECK kalit so'zi ko'rsatilgan maydondan foydalanuvchi predikat ifodalaridan iboratdir.

Misol: Salepeople jadvali Comm ustuniga kiritilayotgan qiymat 1 dan kichik bo'lish sharti.

CREATE TABLE Salepeople
(SNum integer NOT NULL PRIMARY KEY,
SName char(10) NOT NULL UNIQUE,
City char(10),
Comm decimal CHECK (Comm < 1))

CHECK cheklanishidan maydonga ma'lum qiymatlarini kiritishdan himoya qilib, xatolar oldini olish uchun foydalanish mumkin. Masalan mahsulotni sotish shoxobchalariga ega bo'lgan shaharlar faqat London, Barselona, San Xose va Nyu York bo'lsin.

CREATE TABLE Salepeople
(SNum integer NOT NULL PRIMARY KEY,
SName char(10) NOT NULL UNIQUE,
City char(10) CHECK (City IN ("London", "New York", "San Hose", "Barselona")),
Comm decimal CHECK (Comm < 1)

CHECK jadval cheklanishi sifatida kelishi mumkin. Bu shartga bir necha maydon kiritishga imkon beradi.

Masalan:

CREATE TABLE Salepeople
(SNum integer NOT NULL PRIMARY KEY,
SName char(10) NOT NULL UNIQUE,
City char(10),
Comm decimal,
CHECK (Comm < 15 OR City = "Barcelona"))

Ko'zda tutilgan qiymatlarni o'rnatish.

Biror bir maydon uchun qiymat ko'rsatmagan holda jadvalga satr qo'shsangiz, SQL bunday maydonga kiritish uchun ko'zda tutilgan qiymatga ega bo'lishi kerak, aks holda komanda rad etiladi. Eng umumiy ko'zda tutilgan qiymat NULL qiymatdir. CREATE TABLE komandasida ko'zda tutilgan qiymat

DEFAULT operatori orqali, ustun cheklanishi sifatida ko‘rsatiladi. Masalan:

```
CREATE TABLE Salepeople
(SNum integer NOT NULL PRIMARY KEY,
SName char(10) NOT NULL UNIQUE,
City char(10) DEFAULT "New York",
Comm decimal CHECK ( Comm < 1 ))
```

Ma’lumotlar yaxlitligini ta’minlash.

Jadval bir maydonidagi hamma qiymatlar boshqa jadval maydonida aks etsa, birinchi maydon ikkinchisiga ilova qiladi deyiladi. Bu ikki maydon orasidagi bog‘liqlikni ko‘rsatadi. Masalan, buyurtmachilar jadvalida har bir buyurtmachi, sotuvchilar jadvalida o‘ziga biriktirilgan sotuvchiga ilova qiluvchi SNum maydoniga ega. Bir maydon ikkinchisiga ilova qilsa tashqi kalit, u ilova qilayotgan maydon ajdod kalit deyiladi. Buyurtmachilar jadvalidagi SNum maydoni tashqi kalit, sotuvchilar jadvalidagi SNum — ajdod kalitdir.

Tashqi kalit bitta maydondan iborat bo‘lishi shart emas. Birlamchi kalit kabi, tashqi kalit bitta modul sifatida qayta ishlanuvchi bir necha maydonlarga ega bo‘lishi mumkin. Maydon tashqi kalit bo‘lsa ilova qilayotgan jadval bilan ma’lum usulda bog‘liqdir. Tashqi kalit har bir qiymati (satri), ajdod kalitning bitta va faqat bitta qiymatiga (satriga) ilova qilishi kerak. Bu xolda tizim ilovali yaxlit holatda deyiladi.

Shu bilan birga ajdod kalit qiymati tashqi kalit bir necha qiymatlariga ilova qilishi mumkin.

Cheklanish FOREIGN KEY.

SQL ilovali yaxlitlikni FOREIGN KEY yordamida ta’minlaydi. Tashqi kalit vazifasi ajdod kalitda ko‘rsatilmagan qiymatlarni tashqi kalit maydonlariga kiritmaslikdir. FOREIGN KEY cheklanishi sintaksisi:

```
FOREIGN KEY <column list> REFERENCES
<pktable> [<column list>]
```

Birinchi ro‘yxat komanda tomonidan o‘zgartiriluvchi ustunlar ro‘yxatidir. Pktable — bu ajdod kalitli jadval. Ikkinchi ustunlar ro‘yxati bu ajdod kalitni tashkil qiluvchi ustunlardir.

Misol uchun Sotuvchilar jadvaliga ilova qiluvchi tashqi kalit sifatida e’lon qilingan SNum maydoniga ega bo‘lgan Buyurtmachilar jadvalini yaratamiz:

CREATE TABLE Customers
(CNum integer NOT NULL PRIMARY KEY,
CName char(10),
City char(10),
SNum integer,
FOREIGN KEY (SNum) REFERENCES Salepeople (SNum))

Tashqi kalitni ustunlar cheklanishi sifatida berish mumkin. Buning uchun FOREIGN KEY ko‘rinishi — ko‘rsatkichli cheklanish (REFERENCES) qo‘llanadi:

CREATE TABLE Customers
(CNum integer NOT NULL PRIMARY KEY,
CName char(10),
City char(10),
SNum integer REFERENCES Salepeople (SNum))

FOREIGN KEY cheklanishidan jadval yoki ustun cheklanishi sifatida foydalanganda, agar ular PRIMARY KEY cheklanishiga ega bo‘lsa, ajdod kalit ustunlarini ko‘rsatmaslik mumkin.

Kalitlarga cheklanish.

Ilovali yaxlitlikni ta’minlash tashqi kalit yoki ajdod kalit maydonlari qiymatlariga cheklanishlar o‘rnatishni talab qiladi. Ajdod kalit tarkiblangan bo‘lib, tashqi kalit har bir qiymati bitta satrga mos kelishi ta’minlangan bo‘lishi kerak. Bu kalit unikal bo‘lib, bo‘sh (NULL) qiymatlarga ega bo‘lmasligi kerak. Shuning uchun ajdod kalit maydonlari PRIMARY KEY cheklanishiga ega bo‘lishi yoki NOT NULL cheklanishi bilan birga UNIQUE deb e’lon qilinishi kerak.

Tashqi kalit ajdod kalitda mavjud qiymatlarga yoki bo‘sh (NULL) qiymatga ega bo‘lishi mumkin. Boshqa qiymat kiritishga urinish rad etiladi. Tashqi kalitga NOT NULL deb e’lon qilish mumkin, lekin bu maqsadga muvofiq emas. Masalan, siz qaysi sotuvchi mos kelishini bilmasdan oldin buyurtmachini kiritmoqchisiz. Bu holda NULL qiymatdan foydalanib, keyinchalik uni konkret qiymatga almashtirish mumkin.

Cheklanishlar ta’siri.

Tashqi kalit maydonlariga INSERT yoki UPDATE yordamida kiritilayotgan qiymatlar ajdod kalitlariga oldin kiritilgan bo‘lishi kerak. Tashqi kalit ixtyoriy satrini DELETE yordamida o‘chirish mumkin. ANSI ta’rifi bo‘yicha: tashqi kalit

yordamida ilova qilinayotgan ajdod kalit qiymatini o‘chirib yoki o‘zgartirib bo‘lmaydi. Bu shuni bildiradiki, buyurtmalar jadvalida buyurtmalarga ega buyurtmachini, buyurtmachilar jadvalidan o‘chirib bo‘lmaydi. ANSI tarkibiga kirmagan ajdod kalit maydonlarini o‘zgartirish yoki o‘chirish qoidalari mavjud:

1) Cheklangan (RESTRICT) o‘zgartishlar. Siz (ANSI usulida) ajdod kalitlarda cheklangan deb ko‘rsatishingiz yoki man qilishingiz mumkin.

2) Kaskadlanuvchi (CASCADE) o‘zgartishlar. Agarda ajdod kalitda o‘gartish kiritsangiz, tashqi kalitda xuddi shunday o‘zgartishlar avtomatik yuz beradi.

3) Bo‘sh (NULL) o‘zgartishlar. Siz ajdod kalitda o‘zgartirish kiritganingizda tashqi kalit maydonlari avtomatik NULL qiymat oladi (tashqi kalitda NULL qiymat ruxsat etilgan bo‘lsa).

Yuqorida ko‘rsatilgan effektlar UPDATE va DELETE komandalari bajarilganda ajdod kalit o‘zgarishini ko‘rsatadi va quyidagicha aniqlanadi:

CREATE TABLE <table-name >
(<column name> <data type>[(<size>)],
<column name> <data type>[(<size>)],
...
FOREGIGN KEY (<column name>,..) REFERENCES <table name>[(<column name>, ...)]
ON UPDATE [CASCADE|RESTRICT|SET NULL],
ON DELETE [CASCADE|RESTRICT|SET NULL], ...)

Misol. Siz sotuvchi nomerini o‘zgartirmoqchisiz, lekin uning hamma buyurtmachilarini saqlab qolmoqchisiz. Lekin bu sotuvchi firmadan bo‘shab ketsa siz uning buyurtmachilarini boshqa sotuvchiga mahkamlashingiz kerak. Buni bajarish uchun kaskad effektli UPDATE va cheklanishli DELETE berishingiz kerak.

CREATE TABLE Customers
(CNum integer NOT NULL PRIMARY KEY,
CName char(10) NOT NULL,
City char(10),
Rating integer,
SNum integer REFERENCES Salepeople
ON UPDATE CASCADE
ON DELETE RESTRICT)

Agar endi sotuvchilar jadvalidan Peel ni o‘chirmoqchi bo‘lsangiz, buyurtmachilar jadvalida Hoffman va Clemens ning SNum maydonini boshqa tayinlangan sotuvchiga o‘zgartirishi-

ngiz kerak. Boshqa tomondan Peel SNum maydonini 1009 ga o‘zgartirsangiz Hoffman va Clemens ham avtomatik o‘zgaradi.

Tranzaksiyalar (Qachon qilingan o‘zgarishlar doimiy bo‘ladi?).

Komanda yoki komandalar guruhi bajarilgandan so‘ng o‘zgartishlar ma’lumotlar bazasida saqlanib qolishi yoki rad etilishini hal qilishingiz lozim. Bu maqsadda komandalar tranzaktsiya deb ataluvchi guruhlarga biriktiriladi.

Har doim SQL seans boshlaganingizda tranzaksiya ham boshlanadi. Hamma komandalar tranzaksiya qismi hisoblanadi, toki ularni COMMIT yoki ROLLBACK komandasi kiritib tugatmaguningizcha. COMMIT o‘zgarishlarni doimiy qiladi, ROLLBACK bo‘lsa rad qiladi. Yangi tranzaksiya COMMIT yoki ROLLBACK komandasidan so‘ng boshlanadi.

Ko‘pgina realizatsiyalarda siz AUTOCOMMIT parametrini o‘rnatishingiz mumkin. Bu hamma qadamlarni avtomatik eslab qoladi. Xatoga olib keluvchi qadamlar teskarisiga bajariladi. Buni quyidagicha bajarish mumkin: SET AUTOCOMMIT ON; Oldingi holatga quyidagicha qaytish mumkin: SET AUTO-COMMIT OFF;

Ba’zi komandalar, ya’ni ALTER, CREATE, DROP, GRANT, REVOKE, kabi COMMIT ni avtomatik bajaradi.

2.3. MAYDONLARNI KIRITISH, O‘CHIRISH VA O‘ZGARTIRISH

Qiymatlarni kiritish.

Hamma satrlar SQLda INSERT komandasi yordamida kiritiladi. INSERT quyidagi formatlardan biriga ega bo‘lishi mumkin:

INSERT INTO <table name | view name> [(column [,column] ...)]
VALUES (<value> [,<value>] ...);

yoki

INSERT INTO <table name | view name> [(column [,column] ...)]

Ostki so‘rov.

Masalan, sotuvchilar jadvaliga satr kiritish uchun quyidagi shartdan foydalanishingiz mumkin:

INSERT INTO Salepeople
VALUES (11, "Peel", "London", .12);

Siz nom kiritish uchun ustunlar ko‘rsatishingiz mumkin. Bu nomlarni ixtiyoriy tartibda kiritishga imkon beradi. Masalan:

INSERT INTO Salepeople (Sname, Comm, SNum)
VALUES (″Peel″, .12, 11)

E’tibor bering, City ustuni tashlab yuborilgan, chunki unga ko‘zda tutilgan qiymat kiritiladi.

Siz INSERT komandasidan bir jadvaldan qiymat tanlab, so‘rov bilan ishlatish uchun, ikkinchisiga joylashishda foydalanishingiz mumkin. Buning uchun siz VALUES ifodasini (oldingi misoldagi) mos so‘rovga almashtiringiz kerak:

INSERT INTO Londonstaff
SELECT * FROM Salespeople
WHERE City = ″London″

Satrlarni o‘chirish.

Satrlarni jadvaldan DELETE komandasi bilan o‘chirish mumkin. U alohida qiymatlarni emas, faqat satrlarni o‘chiradi. DELETE quyidagi formatga ega:

DELETE FROM <table name | view name>
[WHERE search-condition]

Masalan, Sotuvchilar jadvalidagi hamma satrlarni o‘chirish uchun quyidagi shartni kiritish mumkin:

DELETE FROM Salepeople

Ma’lum satrlarni o‘chirish uchun predikatdan foydalaniladi. Masalan, jadvaldan Axelrod sotuvchini o‘chirish uchun:

DELETE FROM Salepeople
WHERE SNum = 13

Maydon qiymatlarini o‘zgartirish.

Bu o‘zgartirish UPDATE komandasi yordamida bajariladi. Bu komandada UPDATE ifodasidan so‘ng jadval nomi va SET ifodasidan so‘ng ma’lum ustun uchun o‘zgartirish ko‘rsatiladi. UPDATE ikki formatga ega. Ulardan birinchisi:

UPDATE <table name | view name>
SET column = expression [, column = expression] ...
[WHERE search-condition]

bu yerda expression — bu ustun | ifoda | konstanta | o‘zgaruvchi.

Ikkinchi variant:

UPDATE <table name>
SET column = expression, ...

[FROM table-list]
[WHERE search-condition]

Masalan, hamma buyurtmachilar bahosini 200 ga o‘zgartirish uchun quyidagini kiritishingiz mumkin:

UPDATE Customers
SET Rating = 200

Ma’lum satrlarni o‘zgartirish uchun DELETE dagi kabi predikatdan foydalanish kerak. Masalan Peel (SNum=11) sotuvchining hamma buyurtmachilari uchun bir xil o‘zgartirishni quyidagicha kiritish mumkin:

UPDATE Customers
SET Rating = 200
WHERE SNum = 11

SET vergul bilan ajratilgan ixtiyoriy sondagi ustunlarga qiymat tayinlashi mumkin. Ixtiyoriy jadval satrlari uchun qiymat tayinlanishi mumkin, lekin bir vaqtning o‘zida faqat bitta satrga qiymat tayinlanadi. Masalan:

UPDATE Salepeople
SET SName = ″Gibson″, City = ″Boston″, Comm = .10
WHERE SNum = 14

Siz UPDATE komandasining SET jumlasida skalyar ifodalardan o‘zgartirilayotgan maydon ifodasiga qo‘shgan holda foydalanishingiz mumkin. Masalan:

UPDATE Salepeople
SET Comm = Comm * 2

2.4. SELECT SO‘ROV OPERATORI

SELECT operatori MB jadvallaridan natijaviy to‘plam olish uchun mo‘ljallangan ifodadir. Biz SELECT operatori yordamida so‘rov beramiz, u bo‘lsa ma’lumotlar natijaviy to‘plamini qaytaradi. Bu ma’lumotlar jadval shaklida qaytariladi. Bu jadval keyingi SELECT operatori tomonidan qayta ishlanishi mumkin va hokazo.

Operator SQL92 standartiga ko‘ra quyidagi ko‘rinishga ega:

SELECT — ALL ----- s- sxema , ustun ----------
— DISTINCT — ---- * ----------
FROM — sxema, Jadval .. ----------
WHERE — izlash sharti ----------
GROUP BY — sxema, ustun ----------
HAVING — izlash sharti ----------
ORDER BY — tartiblash spetsifikatori ----------

Birinchi qoida: SELECT ifodasi o‘z ichiga albatta FROM ifodasini olishi kerak. Qolgan ifodalar kerak bo‘lsa ishlatiladi.

SELECT ifodasidan so‘ng so‘rovda qaytariluvchi ustunlar ro‘yxati yoziladi.

FROM ifodasidan so‘ng so‘rovni bajarish uchun jadvallar nomi yoziladi.

WHERE ifodasidan so‘ng agar ma’lum satrlarni qaytarish lozim bo‘lsa, izlash sharti yoziladi.

GROUP BY ifodasi guruhlarga ajratilgan natijaviy so‘rov yaratishga imkon beradi.

HAVING ifodasidan guruhlarni qaytarish sharti yoziladi va GROUP BY bilan birga ishlatiladi.

ORDER BY ifodasi ma’lumotlar natijaviy to‘plamini tartiblash yo‘nalishini aniqlaydi.

OFFICES jadvalidagi hamma yozuvlarni qaytaruvchi sodda so‘rov ko‘ramiz.

SELECT * FROM OFFICES

SELECT yordamida ma’lumotlarni tanlash.

SELECT operatori albatta «qaytariluvchi ustunlar ro‘yxati»ni o‘z ichiga olishi kerak, Ya’ni:

SELECT FILED1, FIELD2, FIELD3 ... FROM ...

FILED1, FIELD2, FIELD3 qaytariluvchi ustunlar ro‘yxati bo‘lib, ma’lumotlar ketma-ketligi shu tartibda qaytariladi!

Ya’ni «qaytariluvchi ustunlar ro‘yxati» hisoblanuvchi ustunlar va konstantalarni o‘z ichiga olishi mumkin:

SELECT FILED1, (FIELD2 — FIELD3) "CONST" ... FROM ...

FROM jumlasi «jadval spetsifikatorlari» ya’ni so‘rovni tashkil qiluvchi jadvallar nomini o‘z ichiga oladi. Bu jadvallar so‘rov asosini tashkil qiluvchi jadvallar deyiladi.

Misol: Hamma xizmatchilarning nomlari, ofislari va ishga olish sanalari ro‘yxatini hosil qilish.

SELECT NAME, REP_OFFICE, HIRE_DATE FROM SALESREPS

SELECT operatori qaytaruvchi ustunlar ixtiyoriysi hisoblanuvchi, ya’ni natijada mustaqil ustun sifatida tasvirlanuvchi matematik ifoda bo‘lishi mumkin.

Misol: Har bir ofis uchun shaharlar, regionlar va sotuvlar rejasi qanchaga ortig‘i yoki kami bilan bajarilganligi ro‘yxati.

SELECT CITY, REGION, (SALES-TARGET) FROM OFFICES

Har bir xizmatchi uchun rejadagi sotuvlar xajmini haqiqiy sotuvlar hajmining 3% foiziga oshirish!

SELECT NAME, QUOTA, (QUOTA +((SALES/100)*3)) FROM SALESREPS

Ba'zida ustunlardan biri izlash shartiga bog'liq bo'lmagan qiymat qaytarishi kerak bo'ladi.

Masalan: Har bir shahar uchun sotuvlar hajmlari ro'yxatini chiqaring.

SELECT CITY, "Has sales of", SALES FROM OFFICES

"Has sales of" bu konstantalar ustunidir.

Ba'zida ma'lumotlarni tanlashda qaytariluvchi qiymatlar xosil bo'ladi.

Bu hol yuz bermasligi uchun DISTINCT operatoridan foydalanish lozim. Masalan, quyidagicha:

SELECT DISTINCT MGR FROM OFFICES

SELECT operatori WHERE sharti.

Endi WHERE ifodasidan foydalanib ba'zi so'rovlarni ko'rib chiqamiz: Sotuvlar haqiqiy hajmi rejadan oshgan ofislarni ko'rsating.

SELECT CITY, SALES, TARGET FROM OFFICES WHERE SALES > TARGET

Identifikatori 105 ga teng bo'lgan xizmatchi nomi haqiqiy va rejadagi sotuvlar hajmini ko'rsating:

SELECT SALES, NAME, QUOTA FROM SALESREPS WHERE EMPL_NUM = 105

Agar izlash sharti TRUE bo'lsa, qator natijaviy to'plamga qo'shiladi, agar izlash sharti FALSE bo'lsa, qator natijaviy to'plamga qo'shilmaydi, agar NULL bo'lsa ham natijaviy to'plamdan chiqariladi. O'z ma'nosiga ko'ra WHERE, keraksiz yozuvlarni chiqarib, kerakligini qoldiruvchi filtr sifatida ishlatiladi.

Asosiy izlash shartlari «predikatlar», beshta. Ularni ko'rib chiqamiz:

1. Solishtirish, Ya'ni bir shart natijasi ikkinchisi bilan solishtiriladi. Birinchi so'rov kabi.
2. Qiymatlar diapazoniga tegishlilikni tekshirish. Masalan, berilgan qiymat diapazonga kiradimi, yo'qmi?
3. To'plam elementiligini tekshirish. Masalan, ifoda qiymati to'plamdagi biror qiymat bilan ustma-ust tushadimi?
4. Shablonga moslikni tekshirish. Ustundagi satrli qiymat shablonga mos keladimi?

5. NULL qiymatga tenglikka tekshirish.

Solishtirish amallari maydon va konstantalarni solishtirish amallarini o‘z ichiga olishi mumkin: 1988-yilgacha ishga olingan hamma xizmatchilar nomlarini toping.

SELECT NAME FROM SALESREPS
WHERE HIRE_DATE< TO_DATE("01.06.1988", "DD/MM/YYYY")

TO_DATE("01.06.1988", "DD/MM/YYYY") — PL/SQL Oracle sana bilan ishlash standart funksiyasi.

Yoki arifmetik ifodalarni o‘z ichiga olishi mumkin: Haqiqiy sotuvlar hajmi rejaning 80 foizidan kam bo‘lgan ofislar ro‘yxatini chiqaring.

SELECT CITY, SALES, TARGET FROM OFFICES
WHERE SALES < (0.8 * TARGET)

Ko‘p hollarda izlash birlamchi kaliti bo‘yicha konstantalar bilan solishtirish so‘rovlaridan foydalaniladi. Masalan, shahar telefon tarmog‘i abonenti, axir ikkita bir xil nomerlar mavjud emas-ku.

2.5. MANTIQIY OPERATORLAR

BETWEEN va IN Operatorlari.

BETWEEN ifodasi bu qiymatlar diapazoniga tegishlilikni tekshirishdir. Ifoda sintaksisi quyidagicha:

--- tekshirilayotgan ifoda ------ BETWEEN ---- quyi ifoda AND yuqori ifoda

— NOT -

NOT ifodasi shartni teskarisiga o‘giradi, ya’ni tegishli emas degan ma’noni bildiradi.

Misol: Narxi har xil diapazonga mos keluvchi buyurtmalarni topish.

SELECT ORDER_NUM, AMOUNT
FROM ORDERS
WHERE AMOUNT BETWEEN 20.000 AND 29.999

NOT ifodasi yordamida berilgan diapazonga tegishlilikni tekshirish mumkin, masalan: Sotuvlar haqiqiy hajmlari rejaning 80 dan 120 protsentigacha bo‘lgan diapazonga tushmaydigan xizmatchilar ro‘yxatini chiqarish.

SELECT NAME, SALES, QUOTA
FROM SALESREPS
WHERE SALES NOT BETWEEN (0.8 * QUOTA) AND (1.2 * QUOTA)

Ifoda IN to‘plamga tegishlilikni tekshiradi. Komanda sintaksisi quyidagicha:

-- tekshirilayotgan ifoda ---- IN ---- (— const ---------)
— NOT — ----, --------

1990-yil iyun oyining har xil kunlarida qilingan hamma buyurtmalarni aniqlash.

```
SELECT ORDER_NUM, ORDER_DATE, AMOUNT
FROM ORDERS
WHERE ORDER_DATE IN
(TO_DATE("14.06.1990", "DD/MM/YYYY"),
TO_DATE("08.06.1990", "DD/MM/YYYY"),
TO_DATE("29.06.1990","DD/MM/YYYY"),
TO_DATE("04.06.1990","DD/MM/YYYY"))
```

Sanalar bilan shu tarzda ishlanadi.

To‘rtta konkret xizmatchilar tomonidan olingan hamma buyurtmalarni aniqlash.

```
SELECT ORDER_NUM, REP, AMOUNT
FROM ORDERS
WHERE REP IN (107, 109, 101, 103)
```

NOT IN yordamida diapazonga «tegishli emas»likni tekshirish mumkin.

Operator LIKE

LIKE ifodasi sintaksisi SQL92 standarti bo‘yicha quyidagi ko‘rinishga ega:

--- ustun nomi ---- LIKE (shablon) ----------
NOT ESCAPE
(o‘tkazish nomi)

Sodda so‘rov bajaramiz: «Apelsin» kompaniyasi uchun kredit limitini ko‘rsatish:

```
SELECT COMPANY, CREDIT_LIMIT
FROM CUSTOMERS
WHERE COMPANY = "Apelsin"
```

Quyidagicha "%" shablonli LIKE operatorini qo‘llaymiz:

```
SELECT COMPANY, CREDIT_LIMIT
FROM CUSTOMERS
WHERE COMPANY LIKE "%n"
```

Bu holda LIKE "%n" operatori "n" harfiga tugaydigan hamma yozuvlarni ko‘rsatadi, agar "%" shabloni birinchi kelsa:

```
SELECT COMPANY, CREDIT_LIMIT
FROM CUSTOMERS
WHERE COMPANY LIKE "%gan"
```

Ba'zida "%" shabloni o'rniga "*" belgisi qo'llanadi , masalan MS SQL uchun, c:\>dir *.exe!

Agar faqat bitta simvol ixtiyoriy bo'lsa "_" shabloni qo'llanadi. Masalan:

```
SELECT COMPANY, CREDIT_LIMIT
FROM CUSTOMERS
WHERE COMPANY LIKE "Ap_lsin"
```

Operator IS NULL

SELECT operatori uchun NULL qiymati bilan ishlash qoidalarini konkret misolda ko'ramiz:

Hali ofisga biriktirilmagan xizmatchini topish:

```
SELECT NAME FROM SALESREPS
WHERE REP_OFFICE = NULL
```

SQL quyidagi satrni uchratganda:

```
REP_OFFICE = NULL
```

Quyidagi shartni tekshiradi:

```
NULL = NULL
```

Bunday tekshirish yana NULL qaytaradi! Qiymat tekshiruvchi operator uchun agar natija TRUE bo'lmasa, satr natijaviy to'plamga kirmaydi! Lekin bunday satrlar aslida mavjuddir! Bu holda NULL qiymatiga tekshirish to'g'ri operatorini qo'llash lozim:

```
------ ustunning nomi  IS ------ NULL -----------
                          NOT
```

Qo'llaymiz:Ofisga biriktirilmagan xizmatchini toping.

```
SELECT NAME FROM SALESREPS
WHERE REP_OFFICE IS NULL
```

NOT shartini qo'llash mumkin: Ofisga biriktirilgan hamma xizmatchilarni toping.

```
SELECT NAME FROM SALESREPS
WHERE REP_OFFICE IS NOT NULL
```

WHERE shartida qo'shma operatorlar.

Izlashning «qo'shma» shartlarini ko'rib chiqamiz. WHERE operatorida OR, AND, NOT operatorlari bilan bog'langan bir necha izlash shartlarini qo'llash mumkin. Bu operatorlar sintaksisi quyidagicha:

NOT, OR, AND operatorlarning sintaksisi.

```
(------ WHERE ------  SHART ----------------)
(---- NOT ---)
(---------- AND ----------------)
(---------- OR -----------------)
```

Bu operatorlar yordamida yaratilgan bir necha so‘rovlarni ko‘rib chiqamiz.

Masalan: Sotuvlari haqiqiy hajmi rejadagidan yoki 300.0 $ dan kam bo‘lgan xizmatchilarni aniqlash;

```
SELECT NAME, QUOTA, SALES
FROM SALESREPS
WHERE SALES < QUOTA OR SALES < 300.0
```

Sotuvlari haqiqiy hajmi rejadagidan va 300.0 $ dan kam bo‘lgan xizmatchilarni aniqlash:

```
SELECT NAME, QUOTA, SALES
FROM SALESREPS
WHERE SALES < QUOTA AND SALES < 300.0
```

Sotuvlari haqiqiy hajmi rejadagidan kam, lekin 150.0 $ dan ko‘p bo‘lgan xizmatchilarni aniqlash:

```
SELECT NAME, QUOTA, SALES
FROM SALESREPS
WHERE (SALES < QUOTA) AND (NOT SALES > 150.0)
```

AND ifodasi algebrasi.

Qiymat	Natija
TRUE AND TRUE	-> TRUE
FALSE AND TRUE	-> FALSE
TRUE AND FALSE	-> FALSE
FALSE AND FALSE	->FALSE
NULL AND TRUE	-> NULL
TRUE AND NULL	-> NULL
FALSE AND NULL	->FALSE
NULL AND FALSE	->FALSE
NULL AND NULL	-> NULL

OR ifodasi algebrasi.

Qiymat	Natija
TRUE OR TRUE	-> TRUE
FALSE OR TRUE	-> TRUE
TRUE OR FALSE	-> TRUE
FALSE OR FALSE	-> FALSE
NULL OR TRUE	-> TRUE
TRUE OR NULL	-> TRUE
FALSE OR NULL	-> NULL
NULL OR FALSE	-> NULL
NULL OR NULL	-> NULL

NOT ifodasi algebrasi.

Qiymat	Natija
NOT TRUE	**-> FALSE**
NOT FALSE	**->TRUE**
NOT NULL	**-> NULL**

Qo‘shma izlash operatorlarining har biri o‘z ustivorligiga ega. Eng yuqori ustivorlik NOT ga tegishli, undan so‘ng AND va oxirida OR.

SQL92 standartida IS operatori yordamida ifoda rost, yolg‘on yoki aniqlanmaganligini tekshirish mumkin. Uning sintaksisi quyidagicha:

IS operatori sintaksisi.

----- Solishtirish --------- IS (------ TRUE ----------)
(------ FALSE ----------)
----- Mantiqiy ifoda ----- (---- UNKNOWN --------)

Masalan, quyidagicha yozish mumkin: ((SALES — QUOTA) > 100.000) IS UNKNOWN. Bunday shart SALES yoki QUOTA ustunlari NULL qiymatga ega satrlarni izlashga imkon beradi.

Yozuvlarni tartiblash, ORDER BY jumlasi.

Oldin ko‘rilgan so‘rovlarda natijalar ixtiyoriy tartibda olingan edi. Agar o‘quvchilar ro‘yxatini alfavit tartibida yoki tovarlar narxini kamayish tartibida chiqarish zarur bo‘lsachi? Buning uchun SELECT operatori tarkibida ORDER BY ifodasi ko‘zda tutilgan. Uning sintaksisi:

-------- ORDER BY BY — ustun nomi ---------- .
------ ustun tartib raqami --- ---- ASC ----------
------ DESC ----------
---------- , ----------

Avval quyidagi misolni ko‘ramiz: Har bir ofis uchun sotuvlar haqiqiy hajmlarini regionlar nomlari, har bir regionda esa shaharlar nomlari bo‘yicha alfavit tartibida ko‘rsatish.

```
SELECT CITY, REGION, SALES
FROM OFFICES
ORDER BY REGION, CITY
```

ORDER BY ifodasidan keyin kelgan ustun ASOSIY kalitdir, undan keyingi ustunlar ikkinchi darajali kalitlardir. Yozuvlarni o‘sish hamda kamayish bo‘yicha tartiblash mumkin.

Masalan: Sotuvlari haqiqiy hajmlari kamayish tartibida ofislar ro‘yxatini chiqarish.

SELECT CITY, REGION, SALES
FROM OFFICES
ORDER BY SALES DESC

Sotuvlar hajmlarini DESC predikatini qo‘llab kamayish tartibida chiqaramiz. O‘sish tartibida chiqarish uchun ASC predikati qo‘llanadi. Bu predikat ko‘zda tutilgan bo‘lib, uni ko‘rsatish shart emas. Agar ustun hisoblanuvchi bo‘lib, nomga ega bo‘lmasa uning tartib nomerini ko‘rsatish mumkin.

Masalan, quyidagicha: Sotuvlar haqiqiy va rejadagi hajmlari ayirmasi kamayish tartibida ofislar ro‘yxatini chiqaring.

SELECT CITY, REGION, (SALES — TARGET)
FROM OFFICES
ORDER BY 3 DESC

Shu kabi ORDER BY ifodasida ustunlar nomlari va nomerlari hamda DESC, ASC predikatlarini qo‘llab, murakkab tartiblash shartlarini hosil qilish mumkin.

Masalan: Regionlar nomlari, har bir regionda sotuvlar haqiqiy va rejadagi hajmlari ayirmasi kamayish tartibida ofislar ro‘yxatini chiqaring.

SELECT CITY, REGION, (SALES — TARGET)
FROM OFFICES
ORDER BY REGION ASC, 3 DESC

2.6. BIR NECHA JADVALLAR BILAN ISHLASH

Jadvallarni jamlashtirish.

Jamlashtirish relyatsion ma’lumotlar bazasi operatsiyalaridan biri bo‘lib, jadvallar orasidagi aloqani belgilaydi va ulardan ma’lumotni bitta komanda yordamida ajratishga imkon beradi. Har xil jadvallarda bir xil nomli ustunlar bo‘lishi mumkin bo‘lgani uchun, kerakli ustun uchun jadval nomi prefiksi ishlatiladi.

Jamlashda jadvallar FROM ifodasidan so‘ng ro‘yxat sifatida tasvirlanadi. So‘rov predikati ixtiyoriy jadval ixtiyoriy ustuniga tegishli bo‘lishi mumkin. Jamlashning eng soddasi bu dekart ko‘paytmasi, uni quyidagicha bajarish mumkin:

SELECT Customers.*, Salepeople.*
FROM Salepeople, Customers; *

Lekin bu yerda hosil bo‘lgan jadval keraksiz ma’lumotlarga

ega. Keraksiz satrlarni olib tashlash uchun WHERE jumlasidan foydalaniladi.

Masalan: berilgan shahardagi sotuvchilar va buyurtmachilar ixtiyoriy kombinatsiyasini ko‘rish uchun quyidagini kiritish lozim:

SELECT Customers. CName, Salepeople. SName, Salepeople.City
FROM Salepeople, Customers
WHERE Salepeople.City = Customers.City

Jamlashda SQL bir necha jadval satrlari kombinatsiyasini predikatlar bo‘yicha solishtirishdir. Asosan ma’lumotlar ilovali yaxlitlik asosida tekshirilib, ajratib olinadi.

Misol: har bir sotuvchiga mos keluvchi buyurtmachilar ro‘yxati:

SELECT Customers.CName, Salepeople.SName
FROM Customers, Salepeople
WHERE Salepeople.SNum = Customers.SNum;

Tenglikka asoslangan predikatlardan foydalanuvchi jamlanmalar, tenglik bo‘yicha jamlanma deb atalib, jamlanmalarning eng umuiy ko‘rinishidir. Shu bilan birga ixtiyoriy relyatsion operatordan foydalanish mumkin.

Ichki va tashqi jamlashlar.

Jamlashlar bir jadval satriga ikkinchi jadval satrlarini mos qo‘yishga imkon beradi. Jamlashlar asosiy turi bu ichki jamlashdir. Jadvallarni ichki jamlash ikki jadval usutunlarini tenglashtirishga asoslangandir:

SELECT book, title, author, name
FROM author, book
WHERE book, author = author, id

MySQL jamlashning kuchliroq tip, ya’ni chap tashqi jamlash(yoki tashqi jamlash) dan foydalanishga imkon beradi.

Jamlash bu turining ichki jamlashdan farqi shundaki, natijaga o‘ng jadvalda mos ustunga ega bo‘lmagan chap jadval ustunlari qo‘shiladi. Agar mualliflar va kitoblar misoliga e’tibor bersangiz natijaga ma’lumotlar bazasida kitoblarga ega bo‘lmagan kitoblar kirmagan edi.

Ko‘p hollarda o‘ng jadvalda mosi bo‘lmagan chap jadvaldagi satrlarni chiqarish kerak bo‘ladi. Buni tashqi jamlash yordamida amalga oshirish mumkin:

SELECT book.title, author.name

FROM author
LEFT JOIN book ON book.author = author.id

E'tibor bering: tashqi jamlanmada WHERE o'rniga ON kalit so'zi ishlatiladi.

MySQL tabiiy tashqi jamlashdan *(natural outer join)* foydalanishga imkon beradi. Tabiiy tashqi jamlash ikki jadval ikki ustuni bir xil nom va bir xil tipga ega bo'lgan hamda shu ustundagi qiymatlar teng bo'lgan satrlarni birlashtirishga imkon beradi:

SELECT my_prod.name
FROM my_prod
NATURAL LEFT JOIN their_prod

Jadvallarni o'zi bilan jamlash.

Jadvallarni o'zi bilan jamlash uchun har bir satrning o'zi yoki boshqa satrlar bilan kombinatsiyasini hosil qilishingiz mumkin. So'ngra har bir satr predikat yordamida baholanadi. Bu turdagi jamlash boshqa turdagi jamlashdan farq qilmaydi, farqi ikki jadval bir xildir. Jadvallarni jamlashda qaytariluvchi ustun nomlari oldiga jadval nomi qo'yiladi. Bu usutunlarga so'rovlarda murojaat qilish uchun har xil nomlarga ega bo'lishi kerak. Buning uchun vaqtinchalik nomlar, ya'ni psevdonimlar qo'llanadi. Ular so'rov FROM jumlasida jadval nomidan so'ng bo'shlik qo'yib yoziladi.

Misol: bir xil reytingga ega hamma buyurtmachilar juftlarini topish.

SELECT a.CName, b.CName, a.Rating
FROM Customers a, Customers b
WHERE a.Rating = b.Rating

Bu holda SQL a va b jadvallarni jamlagandek ish tutadi. Yuqorida keltirilgan misolda ortiqcha satrlar mavjud, har bir kombinatsiya uchun ikkita qiymat. Birinchi psevdonimdagi A qiymat ikkinchi psevdonimdagi B qiymat bilan kombinatsiyasi olinadi, so'ngra ikkinchi psevdonimdagi A qiymat birinchi psevdonimdagi B qiymat bilan kombinatsiyasi olinadi.

Har gal satr o'zi bilan solishtiriladi. Buni oldini olishning sodda usuli ikki qiymatga cheklanish kiritishdir, toki birinchi qiymat ikkinchisidan kichik bo'lsin yoki alifbo bo'yicha oldin kelsin. Bu predikatni asimmetrik qiladi, natijada xuddi shu qiymatlar teskari tartibda olinmaydi.

Misol:

SELECT a.CName, b.CName, a.Rating
FROM Customers a, Customers b
WHERE a.Rating = b.Rating
AND a.CName < b.CName

Bu misolda agar birinchi kombinatsiya ikkinchi shartni qanoatlantirsa u chiqariladi, lekin teskari kombinatsiya bu shartni qanoatlantirmaydi va aksincha. Siz SELECT ifodasida yoki so‘rovning FROM jumlasida keltirilgan har bir psevdonim yoki jadvalni ishlatishingiz shart emas. Siz har xil jadvallar, hamda bitta jadval har psevdonimlaridan iborat jamlanma yaratishingiz mumkin.

Sodda joylashtirilgan ostki so‘rovlar.

SQL yordamida so‘rovlarni bir birining ichiga joylashtirishingiz mumkin. Odatda ichki so‘rov qiymat hosil qiladi va bu qiymat tashqi predikat tomonidan tekshirilib, to‘g‘ri yoki noto‘g‘riligi tekshiriladi.

Misol: bizga sotuvchi nomi ma’lum: Motika, lekin biz SNum maydoni qiymatini bilmaymiz va Buyurtmachilar jadvalidan hamma buyurtmalarni ajratib olmoqchimiz. Buni quyidagicha amalga oshirish mumkin:

SELECT * FROM Orders
WHERE SNum =
(SELECT SNum FROM Salepeople
WHERE SName = "Motika");

Avval ichki so‘rov bajariladi, so‘ngra uning natijasi tashqi so‘rovni hosil qilish uchun ishlatiladi (SNum ostki so‘rov natijasi bilan solishtiriladi).

Ostki so‘rov bitta ustun tanlashi lozim, bu ustun qiymatlari tipi predikatda solishtiriladigan qiymat tipi bilan bir xil bo‘lishi kerak. Siz ba’zi hollarda ostki so‘rov bitta qiymat hosil qilishi uchun DISTINCT operatoridan foydalanishingiz mumkin.

Misol: Hoffman (CNum=21) ga xizmat ko‘rsatuvchi sotuvchilar hamma buyurtmalarini topish lozim bo‘lsin.

SELECT * FROM Orders
WHERE SNum = (SELECT DISTINCT SNum FROM Orders WHERE CNum = 21)

Bu holda ostki so‘rov faqat bitta 11 qiymat chiqaradi, lekin umumiy holda bir necha qiymatlar bo‘lishi mumkin va ular ichidan DISTINCT faqat bittasini tanlaydi.

Ixtiyoriy sondagi satrlar uchun avtomatik ravishda bitta qiymat hosil qiluvchi funksiya turi — agregat funksiya bo‘lib, undan ostki so‘rovda foydalanish mumkin.

Masalan, siz summasi 4 oktabrdagi bajarilishi lozim bo‘lgan buyurtmalar summasi o‘rta qiymatidan yuqori bo‘lgan hamma buyurtmalarni ko‘rmoqchisiz:

SELECT * FROM Orders
WHERE AMT >
(SELECT AVG (AMT) FROM Orders
WHERE ODate = ″1990/10/04″)

Shuni nazarda tutish kerakki, guruhlangan agregat funksiyalar GROUP BY ifodasi terminlarida aniqlangan agregat funksiyalar bo‘lsa ko‘p qiymatlar hosil qilishi mumkin.

Agar ostki so‘rov IN operatoridan foydalanilsa, ixtiyoriy sondagi satrlar hosil qilish mumkin.

Misol: Londondagi sotuvchilar uchun hamma buyurtmalarni ko‘rsatish.

SELECT * FROM Orders
WHERE SNum IN
(SELECT SNum FROM Salepeople
WHERE City = ″London″)

Bu natijani jamlanma orqali hosil qilish mumkin. Lekin odatda ostki so‘rovli so‘rovlar tezroq bajariladi. Siz ostki so‘rov SELECT jumlasida ustunga asoslangan ifodadan foydalanishingiz mumkin. Bu relyatsion operatorlar yordamida yoki IN yordamida amalga oshirilishi mumkin. Siz ostki so‘rovlarni HAVING ichida ishlatishingiz mumkin. Bu ostki so‘rovlar agar ko‘p qiymatlar qaytarmasa xususiy agregat funksiyalaridan yoki GROUP BY yoki HAVING operatorlaridan foydalanishi mumkin.

Misol:

SELECT Rating, COUNT (DISTINCT CNum) FROM Customers
GROUP BY Rating
HAVING Rating >
(SELECT AVG (Rating) FROM Customers
WHERE City = ″San Hose″);

Bu komanda San Hose dagi baholari o‘rtachadan yuqori bo‘lgan buyurtmachilarni aniqlaydi.

Korrellangan (mutanosib) joylashtirilgan ostki so‘rovlar.

SQL tilida ostki so‘rovlardan foydalanilganda tashqi so‘rov

FROM qismidagi ichki so‘rovga mutanosib so‘rov yordamida murojaat qilishingiz mumkin. Bu holda ostki so‘rov asosiy so‘rov har bir satri uchun bir martadan bajariladi.

Misol:

3-oktabrda buyurtma bergan hamma buyurtmachilarni toping.

```
SELECT * FROM Customers a
WHERE "1990/10/03" IN
(SELECT ODate FROM Orders b
WHERE a.CNum = b.CNum )
```

Bu misolda tashqi so‘rovning CNum maydoni o‘zgargani uchun ichki so‘rov tashqi so‘rovning har bir satri uchun bajarilishi kerak. Ichki so‘rov bajarilishini talab qiladigan tashqi so‘rov satri joriy satr — kandidat deyiladi. Mutanosib ostki so‘rov bilan bajariladigan baholash protsedurasi quyidagicha:

1. Tashqi so‘rovda nomlangan jadvaldan satrni tanlash. Bu kelajak satr — kandidat.

2. Tashqi so‘rov FROM jumlasida nomlangan psevdonimda bu satr — kandidat qiymatlarini saqlab qo‘yish.

3. Ostki so‘rovni bajarish. Tashqi so‘rov uchun berilgan psevdonim topilgan hamma joyda joriy satr-kandidat qiymatidan foydalanish. Tashqi so‘rov satr-kandidatlari qiymatlaridan foydalanish, tashqi ilova deyiladi.

4. Tashqi so‘rov predikatini 3 qadamda bajariluvchi ostki so‘rov natijalari asosida baholash. U chiqarish uchun satr-kandidat tanlanishini belgilaydi.

5. Jadval keyingi satr-kandidatlari uchun protsedurani qaytarish va shu tarzda toki hamma jadval satrlari tekshirilib bo‘lmaguncha.

Yuqoridagi misolda SQL quyidagi protsedurani amalga oshiradi:

1. U buyurtmachilar jadvalidan Hoffman satrini tanlaydi.

2. Bu satrni joriy satr-kandidat sifatida a — psevdonim bilan saqlaydi.

3. So‘ngra ostki so‘rovni bajaradi. Ostki so‘rov CNum maydonning qiymati a.CNum qiymatiga teng satrlarni topish uchun Buyurtmachilar jadvali hamma satrlarini ko‘rib chiqadi. Hozir a.CNum qiymati 21 ga, ya’ni Hoffman satrining CNum maydoni qiymatiga teng. Shundan so‘ng shu satrlarning ODate maydonlari qiymatlari to‘plamini hosil qiladi.

4. Shundan so‘ng asosiy so‘rov predikatida 3 oktabrdagi

qiymat shu to‘plamga tegishliligini tekshiradi. Agar bu rost bo‘lsa Hoffman satrini chiqarish uchun tanlaydi.

5. Shundan so‘ng u butun protsedurani Giovanni satrini satr — kandidat sifatida foydalanib qaytaradi va toki Buyurtmachilar hamma satri tekshirilib bo‘lmaguncha saqlab qo‘yadi.

Ba’zida xatolarni topish uchun maxsus yaratilgan so‘rovlardan foydalanish kerak bo‘ladi.

Misol: Quyidagi so‘rov Buyurtmachilar jadvalini ko‘rib chiqib SNum va CNum mos kelishini tekshiradi va mos bo‘lmagan satrlarni chiqaradi.

```
SELECT * FROM Orders main
WHERE NOT SNum =
(SELECT SNum FROM Customers
WHERE CNum = main.CNum )
```

Asosiy so‘rov asoslangan jadvalga asoslanuvchi mutanosib so‘rovdan foydalanishingiz mumkin.

Misol: sotib olishlar buyurtmachilari uchun o‘rta qiymatdan yuqori bo‘lgan hamma buyurtmalarni topish.

```
SELECT * FROM Orders a
WHERE AMT >
(SELECT AVG (AMT) FROM Orders b
WHERE b.CNum = a.CNum )
```

HAVING operatoridan ostki so‘rovlarda foydalanilganidek mutanosib ostki so‘rovlarda ham foydalanish mumkin.

HAVING ifodasida mutanosib ostki so‘rovdan foydalanganda HAVING o‘zida ishlatilishi mumkin bo‘lgan pozitsiyalarga tashqi ilovalarni cheklab qo‘yishingiz kerak. Chunki HAVING ifodasidafaqat agregat SELECT ifodasida ko‘rsatilgan funksiyalardan yoki GROUP BY ifodasida ko‘rsatilgan maydonlardan foydalanish mumkin. Ulardan siz tashqi ilova sifatida foydalanishingiz mumkin. Buning sababi shuki, HAVING tashqi so‘rovdagi satrlar uchun emas guruhlar uchun baholanadi. Shuning uchun ostki so‘rov bir marta satr uchun emas, guruh uchun bajariladi.

Misol: Buyurtmalar jadvalidagi sotib olishlar summalarini sanalar bo‘yicha guruhlab summasini hisoblash kerak bo‘lsin. Shu bilan birga summa maksimal summadan kamida 2000.00 ga ko‘p bo‘lmagan sanalarni chiqarib tashlash kerak bo‘lsin:

```
SELECT ODate, SUM (AMT) FROM Orders a
GROUP BY ODate
```

HAVING SUM (AMT) >
(SELECT 2000.00 + MAX (AMT) FROM Orders b
WHERE a.ODate = b.ODate)

Ostki so‘rov asosiy so‘rovning ko‘rilayotgan agregat guruhi sanasiga teng sanaga ega hamma satrlar uchun MAX qiymat hisoblaydi. Bu WHERE ifodasidan foydalanib bajarilishi lozim. Ostki so‘rovning o‘zi GROUP BY yoki HAVING operatorlarini ishlatmasligi kerak.

EXISTS operatoridan foydalanish.

EXISTS — bu “TRUE” yoki “FALSE” qaytaruvchi operatordir. Bu shuni bildiradiki, u predikatda avtonom yoki mantiqiy operatorlar AND, OR, va NOT yordamida tuzilgan mantiqiy ifodalar bilan kombinatsiya qilingan holda ishlatilishi mumkin. U agar ixtiyoriy natija hosil qilsa ostki so‘rovni “TRUE” deb baholaydi va hech qanday natija hosil qilmasa “FALSE” deb baholaydi.

Misol: Agar buyurtmachilardan juda bo‘lmasa bittasi San Hose shahrida yashasa, buyurtmachilar jadvalidagi ma’lumotlarni chiqaring.

SELECT CNum, CName, City FROM Customers
WHERE EXISTS
(SELECT * FROM Customers
WHERE City = ″San Hose″)

EXISTS ni faqat sodda ostki so‘rov bilan emas, mutanosib so‘rov bilan ishlatish mumkin. Bu holda EXISTS ichki ostki so‘rovni tashqining har bir satri uchun tekshiradi.

ALL, ANY, SOME operatorlaridan foydalanish.

ANY, ALL, va SOME ostki so‘rovlarni argument sifatida qabul qiluvchi EXISTS operatorni eslatadi, lekin relyatsion operatorlar bilan birga ishlatilishiga ko‘ra farq qiladi. Bu tomondan ular ostki so‘rovlarga qo‘llaniluvchi IN operatorini eslatadi, lekin undan farqli faqat ostki so‘rovlar bilan ishlashadi. SOME va ANY operatorlari o‘zaro almashinuvchan.

Misol: bir shaharda joylashgan sotuvchilar bilan buyurtmachilarni topish uchun ANY operatoridan foydalanish.

SELECT * FROM Salepeople
WHERE City = ANY (SELECT City FROM Customers)

Operator ANY ostki so‘rov chiqargan hamma qiymatlarni oladi (bu misol uchun — Buyurtmachilar jadvalidagi hamma

City qiymatlari) va agar ularning ixtiyoriysi (ANY) tashqi so‘rov satridagi shahar qiymatiga teng bo‘lsa rost deb baholaydi. ANY operatori o‘rniga IN yoki EXISTS ishlatish mumkin, lekin ANY “=“ operatordan boshqa relyatsionn operatorlarni ishlatishi mumkin. Misol: hamma sotuvchilarni alifbo bo‘yicha kelgan buyurtmachilari bilan birga topish.

SELECT * FROM Salepeople
WHERE SName < ANY (SELECT CName FROM Customers)

ANY to‘la bir qiymatli emas. Misol: Rimdagi buyurtmachilarga ko‘ra yuqori reytinga ega buyurtmachilarni topish.

SELECT * FROM Customers
WHERE Rating > ANY (SELECT Rating FROM Customers WHERE City = ″Rome″)

Ingliz tilida ixtiyoriysidan katta ″(bu yerda City = Rome)″ baholash quyidagicha talqin qilinadi, bu baholash qiymati har bir City = Rome holdagi baholash qiymatidan katta bo‘lishi kerak. SQL tilida ANY operatoridan foydalanilganda bunday emas. ANY agar ostki so‘rov shartga mos keluvchi ixtiyoriy qiymat topsa to‘g‘ri deb baholanadi. Yuqorida ko‘rilgan misol 300 va 200 baholi hamma buyurtmachilarni topadi, chunki 300>200 Rimdagi Giovanni uchun va 200>100 Rimdagi Pereira uchun.

Soddaroq qilib aytganda, < ANY ifodasi eng katta tanlangan qiymatdan kichik qiymatni, > ANY — eng kichik tanlangan qiymatdan katta qiymatni bildiradi.

ALL yordamida, predikat rost hisoblanadi, ostki so‘rov tanlagan har bir qiymat tashqi so‘rov predikatidagi shartga mos kelsa.

Misol: Rimdagi har bir buyurtmachidan baholari yuqori bo‘lgan buyurtmachilarni chiqaring.

SELECT * FROM Customers
WHERE Rating > ALL (SELECT Rating FROM Customers WHERE City = ″Rome″)

Bu operator Rimdagi hamma buyurtmachilar baxolari qiymatlarini tekshiradi. Shundan so‘ng Rimdagi hamma buyurtmachilardan bahosi yuqori bo‘lgan buyurtmachilarni topadi. Rimda eng yuqori baho — Giovanni (200). Demak 200 dan yuqori qiymatlar olinadi.

ANY operatori uchun bo‘lgani kabi ALL operatori uchun ham IN va EXISTS yordamida alternativ konstruksiyalar yaratish mumkin.

ALL asosan tengsizliklar bilan ishlatiladi, chunki qiymat agar hamma natijalar bir xil bo‘lsa «hammasi uchun teng» ostki so‘rov natijasi bo‘lishi mumkin. SQL da < > ALL ifoda aslida ostki so‘rov natijasining «hech qaysisiga teng emas» ma’noni bildiradi. Boshqacha qilib aytganda, agar berilgan qiymat ostki so‘rov natijalari orasida topilmagan bo‘lsa predikat — rost. Agar oldingi misolda tenglik tengsizlikka almashtirilsa, reytingi 300 ga teng bo‘lgan hamma buyurtmachilar chiqariladi, chunki ostki so‘rov 100 va 200 ga teng reytinglarni topgan.

ALL va ANY — orasidagi asosiy farq, ostki so‘rov hech qanday natija qaytarmagan holatda ko‘rinadi. Bu holda ALL — avtomatik “TRUE” ga teng, ANY bo‘lsa avtomatik (“FALSE”) ga teng.

Misol: Buyurtmachilar jadvalining hammasini chiqarish

SELECT * FROM Customers
WHERE Rating > ALL (SELECT Rating FROM Customers WHERE City = ″Boston″)

Ko‘rsatilgan operatorlar bilan ishlashda NULL qiymatlar ma’lum muammolarni keltirib chiqaradi. SQL predikatda solishtirayotgan qiymatlardan biri bo‘sh (NULL) qiymat bo‘lsa, natija noaniqdir. Noaniq predikat, noto‘g‘ri predikatga o‘xshash, shuning uchun satr tashlab yuboriladi.

UNION ifodasidan foydalanish.

UNION ifodasi bir yoki bir necha SQL so‘rovlar natijasini birlashtirishga imkon beradi.

Misol: Londonda joylashgan hamma sotuvchilar va buyurtmachilarni bitta jadvalda chiqaring.

SELECT SNum, SName FROM Salepeople
WHERE City = ″London″
UNION
SELECT CNum, CName FROM Customers
WHERE City = ″London″

Foydali jamlanmalardan biri ikki so‘rovnni jamlashda ikkinchi so‘rov birinchi so‘rov chiqarib tashlagan satrlarni tanlashidir. Bu tashqi jamlanma deyiladi.

Misol: O‘z shaharlarida buyurtmachilarga ega yoki ega emasligini ko‘rsatgan holda hamma sotuvchilarni chiqarish.

SELECT Salepeople.SNum, SName, CName, Comm FROM Salepeople, Customers WHERE Salepeople.City = Customers.City

UNION
SELECT SNum, SName, "NO MATCH", Comm FROM Salepeople WHERE NOT City = ANY (SELECT City FROM Customers) ORDER BY 2 DESC

2.7. SO‘ROVLARDA GURUHLASH VA FUNKSIYALAR

Agregat funksiyalar qo‘llanishi.

Agregat (yoki STATIK) funksiyalar sonli yoki hisoblanuvchi ustunlar bilan ishlaydi. Agregat funksiya argumenti butun ustun bo‘lib, bitta qiymat qaytaradi.

Bu funksiyalarni ko‘rib chiqamiz:

- SUM() – Ustundagi hamma qiymatlar summasini hisoblaydi.
- AVG() – Ustundagi hamma qiymatlar o‘rtasi qiymatini hisoblaydi.
- MIN() – Ustundagi hamma qiymatlar eng kichigini aniqlaydi.
- MAX() – Ustundagi hamma qiymatlar eng kattasini aniqlaydi.
- COUNT() – Ustundagi qiymatlar sonini hisoblaydi.
- COUNT(*) – So‘rov natijalari jadvalidagi satrlar sonini hisoblaydi.

Agregatlash argumenti bo‘lib ustun nomidan tashqari ixtiyoriy matematik ifoda xizmat qilishi mumkin. Misol uchun quyidagi so‘rovda: Sizning kompaniyangizda reja bajarilishining o‘rtacha protsenti qancha?

SELECT AVG(100 * (SALES/QUOTA))
FROM SALESREPS

Yana bir shakl: Siznining kompaniyangizda reja bajarilishining o‘rtacha protsenti qancha?

SELECT AVG(100 * (SALES/QUOTA)) PROCENT
FROM SALESREPS

Bu holda ustun nomi ma’noliroq, lekin bu asosiysi emas. Ustunlar summasini hisoblab ko‘ramiz. SUM() funksiyasini qo‘llaymiz, ustun sonli bo‘lishi kerak. Masalan, quyidagicha: Kompaniya xizmatchilari sotuvlar hajmi rejadagi va haqiqiy o‘rta qiymati qanchaga teng?

SELECT SUM(QUOTA), SUM(SALES)
FROM SALESREPS

AVG() agregatlash funksiyasiga yana bir necha sodda misol-

larni ko‘ramiz. Masalan: «ACI» ishlab chiqaruvchi mollari o‘rtacha narxini hisoblang.

```
SELECT AVG(PRICE)
FROM PRODUCTS
WHERE MFR_ID = "ACI"
```

Ekstremumlarni topish funksiyalari, yani MIN(), MAX() funksiyalarini ko‘ramiz.Bu funksiyalar sonli ustunlar, sanalar va satrli o‘zgaruvchilar bilan ishlaydi. Eng sodda qo‘llanishi sonlar bilan ishlash.

Masalan quyidagicha so‘rov beramiz: Eng ko‘p va kam sotuvlar rejadagi hajmi?

```
SELECT MIN(QUOTA), MAX(QUOTA)
FROM SALESREPS
```

Bu sonlarni o‘z ichiga olgan ustunlardir. Yana bir so‘rov beramiz: Bazadagi buyurtmalarning ichida eng oldin berilgan so‘rov sanasi?

```
SELECT MIN(ORDER_DATE)
FROM ORDERS
```

Satrlar bilan ishlaganda har xil SQL serverlardagi kodirovkalar har xil natija berishi mumkin. Yozuvlar sonini sanash uchun COUNT() qo‘llanadi. Bu funksiya son qiymat qaytaradi.

Masalan: Kompaniyamiz mijozlari soni nechta?

```
SELECT COUNT(CUST_NUM)
FROM CUSTOMERS
```

Yana bir so‘rov: Qancha xizmatchi rejani ortig‘i bilan bajardi?

```
SELECT COUNT(NAME)
FROM SALESREPS
WHERE SALES > QUOTA
```

COUNT(*) funksiyasi qiymatlar sonini emas, satrlar sonini hisoblaydi. Quyidagicha yozish mumkin:

```
SELECT COUNT(*)
FROM ORDERS
WHERE AMOUNT > 250
```

NULL qiymat va agregat funksiyalar.

Ustun qiymati NULL bo‘lsa AVG(), MIN(), MAX(), SUM(), COUNT() funksiyalari qanday qiymat qaytaradi? ANSI/ISO qoidalariga ko‘ra «agregat funksiyalar NULL qiymatni e’tiborga olmaydi». Quyidagi so‘rovni ko‘ramiz:

```
SELECT COUNT(*), COUNT(SALES), COUNT
(QUOTA)
```

FROM SALESREPS

Jadval bitta, lekin so‘rovdagi qiymatlar har xil. Chunki QUOTA maydoni — NULL qiymatni o‘z ichiga oladi. COUNT funksiyasi COUNT(maydon) ko‘rinishda bo‘lsa NULL qiymatni e’tiborga olmaydi, COUNT(*) bo‘lsa satrlar umumiy sonini hisoblaydi. MIN(), MAX() funksiyalari ham NULL qiymatni e’tiborga olmaydi, lekin AVG(), SUM() — NULL qiymat mavjud bo‘lsa chalkashtiradi. Masalan, quyidagi so‘rov:

SELECT SUM(SALES), SUM(QUOTA), (SUM-(SALES) — SUM(QUOTA)), (SUM(SALES — QUOTA))
FROM SALESREPS

(SUM(SALES)-SUM(QUOTA)) va (SUM(SALES-QUOTA)) ifodalari agar QUOTA maydoni NULL qiymatga ega bo‘lsa har xil qiymat qaytaradi. Ya’ni ifoda SUM(ustun qiymati — NULL) yana NULL qaytaradi.

Shunday qilib:

1. Agar ustundagi qiymatlardan biri NULL ga teng bo‘lsa, funksiya natijasini hisoblashda ular tashlab yuboriladi.

2. Agar ustundagi hamma qiymatlar NULL ga teng bo‘lsa, AVG(), SUM(), MIN(), MAX() funksiyalari NULL qaytaradi. Funksiya COUNT() nol qaytaradi.

3. Agar ustunda qiymatlar bo‘lmasa (ya’ni ustun bo‘sh), AVG(), SUM(), MIN(), MAX() funksiyalari NULL qaytaradi. Funksiya COUNT()nol qaytaradi.

4. Funksiya COUNT(*) satrlar sonini hisoblaydi va ustunda NULL qiymat bor-yo‘qligiga bog‘liq emas. Agar ustunda satrlar bo‘lmasa, bu funksiya nol qaytaradi.

DISTINCT funksiyasini agregat funksiyalar bilan birga ishlatish mumkin. Masalan, quyidagi so‘rovlarda:

1. Kompaniyamizda qancha har xil raportlar nomlari mavjud?

SELECT COUNT(DISTINCT TITLE)
FROM SALESREPS

DISTINCT va agregatlar ishlashda quyidagi qoidalar mavjud. Agar siz DISTINCT va agregat funksiyani ishlatsangiz uning argumenti faqat ustun nomi bo‘lishi mumkin, ifoda argument bo‘lolmaydi. MIN(), MAX() funksiyalarida DISTINCT ishlatish ma’nosi yo‘q. COUNT() funksiyasida DISTINCT ishlatiladi, lekin kam hollarda. COUNT(*) funktsiyasiga umuman DISTINCT qo‘llab bo‘lmaydi, chunki u satrlar sonini hisoblay-

di. Bitta so‘rovda DISTINCT faqat bir marta qo‘llanishi mumkin! Agarda u agregat funksiya argumenti sifatida qo‘llanilsa,boshqa argument bilan qo‘llash mumkin emas.

Agregatlar va ma’lumotlarni guruhlash.

Agregat funksiyalar jadval uchun natijaviy satr hosil qiladi. Masalan: Buyurtma o‘rtacha narxi qancha?

SELECT AVG(AMOUNT)
FROM ORDERS

Masalan, oraliq natijani topish lozim bo‘lsin. Bu holda guruhlanishli so‘rov yordam beradi. Ya’ni SELECT operatorining GROUP BY ifodasi. Avval GROUP BY ifodasi qatnashgan quyidagi so‘rovni ko‘ramiz: Har bir xizmatchi uchsun buyurtma o‘rtacha narxi qancha?

SELECT REP, AVG(AMOUNT)
FROM ORDERS
GROUP BY REP

REP maydoni bu holda guruhlash maydonidir, Ya’ni REP maydonning hamma qiymatlari guruhlarga ajratiladi va har bir guruh uchun AVG(AMOUNT) ifodasi hisoblanadi. Ya’ni quyidagilar bajariladi:

1. So‘rovlar har bir xizmatchaga bittadan guruhga ajratiladi.Har bir guruhda REP maydoni bir xil qiymatga ega.

2. Har bir guruh uchun guruhga kiruvchi hamma satrlar bo‘yicha AMOUNT ustuni o‘rta qiymati hisoblanadi va bitta natijaviy satr hosil qilinadi. Bu qator guruh uchun REP ustuni qiymati va shu guruh uchun so‘rov o‘rta qiymatini o‘z ichiga oladi.

Shunday qilib, GROUP BY ifodasi qo‘llanilgan so‘rov, «GURUHLANISHLI SO‘ROV» deb ataladi. Shu ifodadan keyin kelgan ustun «guruhlash ustuni» deyiladi. Yana bir necha guruhlanishli so‘rovlarni ko‘rib chiqamiz.

Har bir ofis uchun sotuvlarning rejalashtirilgan hajmi diapazoni qancha?

SELECT REP_OFFICE, MIN(QUOTA), MAX(QUOTA)
FROM SALESREPS
GROUP BY REP_OFFICE

Yana bir so‘rov: Har bir ofisda qancha xizmatchi ishlaydi?

SELECT REP_OFFICE, COUNT(*)
FROM SALESREPS
GROUP BY REP_OFFICE

Yana bir guruhlanishli qiziqarli so‘rov: Har bir xizmatchi nechta mijozga xizmat ko‘rsatadi?

```
SELECT COUNT(DISTINCT CUST_NUM), "CUSTOMERS FOR SALESREPS", CUST_REP
FROM CUSTOMERS
GROUP BY CUST_REP
```

Bu yerda "CUSTOMERS FOR SALESREPS" psevdomaydonning ishlatilishiga e’tibor bering. So‘rov natijalarini bir nechta ustun bo‘yicha guruhlash mumkin. Masalan, quyidagicha:

Har bir xizmatchi uchun har bir klient bo‘yicha buyurtmalar umumiy sonini hisoblash.

```
SELECT REP, CUST, SUM(AMOUNT)
FROM ORDERS
GROUP BY REP, CUST
```

Lekin ikki ustun bo‘yicha guruhlashda natijalar ikki darajasiga ega guruhlar va ostki guruhlar yaratish mumkin emas. Lekin tartiblashni qo‘llash mumkin. Shu bilan birga GROUP BY ishlatilganda so‘rov natijalari avtomatik tartiblanadi. Quyidagi so‘rovni ko‘ramiz:

Har bir xizmatchi uchun har bir klient bo‘yicha buyurtmalar umumiy sonini hisoblash; so‘rov natijalarini klientlar va xizmatchilar bo‘yicha tartiblash.

```
SELECT REP, CUST, SUM(AMOUNT)
FROM ORDERS
GROUP BY REP, CUST
ORDER BY REP, CUST
```

Shunday qilib GROUP BY ifodasi SELECT ni guruhlarni qayta ishlashga majbur qiladi.

Odatda guruhlanishli so‘rovlar qaytaruvchi ustunlarga guruhlash ustuni va agregat funksiya kiradi. Agar agregat ko‘rsatilmasa GROUP BY dan foydalanmasdan DISTINCT ifodasi-dan foydalanish yetarli. Agar so‘rovga guruhlash ustuni qo‘shilmasa, u yoki bu satr qaysi guruhga tegishliligini aniqlash mumkin emas. Shu kabi SQL92 guruhlanishli so‘rovlarni tahlil qilishda birlamchi va ikkilamchi kalitlar haqidagi ma’lumot ishlatilmaydi.

Har bir xizmatchi uchun buyurtmalar umumiy sonini hisoblash.

```
SELECT EMPL_NUM, NAME, SUM(AMOUNT)
FROM ORDERS, SALESREPS
WHERE REP = EMPL_NUM
```

GROUP BY EMPL_NUM, NAME

Yana soddaroq shakl:

Har bir xizmatchi uchun buyurtmalar umumiy sonini hisoblash.

SELECT NAME, SUM(AMOUNT)
FROM ORDERS, SALESREPS
WHERE REP = EMPL_NUM
GROUP BY NAME

Agar guruhlash maydonlaridan birida NULL qiymat mavjud bo‘lsa qaysi guruhga tegishli bo‘ladi? WHERE ifodasida NULL va NULL tenglikka solishtirish natijasi yana NULL beradi. Shuning uchun ANSI/ISO standartida GROUP BY ifodasida NULL qiymatlar teng deb qabul qilingan.

Guruhlash va HAVING yordamida ajratish.

Shart bo‘yicha satrlarni ajratish uchun WHERE ifodasidan foydalangan edik. Shart bo‘yicha guruhlarni ajratish uchun HAVING operatori mavjuddir. Uning sintaksisi WHERE operatori bilan bir xil va ulardan birgalikda foydalanish mumkin. Quyidagi so‘rovni ko‘ramiz:

Buyurtmalar umumiy narxi 300$ dan ortiq xizmatchilar uchun buyurtma o‘rtacha narxi qanchaga teng?

SELECT REP, AVG(AMOUNT)
FROM ORDERS
GROUP BY REP
HAVING SUM(AMOUNT) > 300

Ko‘rinib turibdiki, HAVING SUM(AMOUNT) > 300 ifodasi satrlarni guruhlash sharti sifatida kelmoqda.

Yana bir misol ko‘raylik: Ikki va undan ortiq xizmatchiga ega har bir ofisning hamma xizmatchilari uchun rejadagi va haqiqiy sotuvlar umumiy hajmini hisoblash.

SELECT CITY, SUM(QUOTA), SUM (SALESREPS. SALES)
FROM OFFICES, SALESREPS
WHERE OFFICE = REP_OFFICE
GROUP BY CITY
HAVING COUNT(*) >= 2

Bu misolda WHERE va HAVING ifodalari o‘z funksiyalarini bajaradilar. Shunga e’tibor berish kerakki, HAVING ifodasida agregat funksiyalardan foydalaniladi, So‘rov bajarilishini ko‘ramiz:

1. OFFICES va SALESREPS jadvallari xizmatchi yashaydigan shaharni topish uchun qo‘shiladilar.

2. Qo‘shilgan jadval satrlarlari ofislar bo‘yicha guruhlanadilar.

3. Ikkidan kam satrga ega guruhlar tashlab yuboriladi. Ular HAVING ifodasi talabiga javob bermaydilar.

4. Har bir guruh uchun haqiqiy va rejadagi sotuvlar hajmlari hisoblanadi.

Murakkabroq misolni ko‘ramiz:

Har bir tovar nomi uchun narxi, ombordagi soni va buyurtma berilganlar umumiy sonini ko‘rsating, agar uning uchun buyurtma berilganlar umumiy soni ombordagi umumiy sonining 75 foizidan ko‘p bo‘lsa.

```
SELECT DESCRIPTION, PRICE, QTY_ON_HAND,
SUM(QTY)
FROM PRODUCTS, ORDERS
WHERE MFR = MFR_ID
GROUP BY MFR_ID, PRODUCT_ID, DESCRIPTION,
PRICE, QTY_ON_HAND
HAVING SUM(QTY) > (0.75 * QTY_ON_HAND)
ORDER BY QTY_ON_HAND DESC
```

HAVING uchung qo‘shimcha chegaralar mavjuddir. Bu ifoda juda bo‘lmasa bitta agregat funksiyani o‘z ichiga olishi kerak. Chunki WHERE alohida satrlarga HAVING satrlar guruhlariga qo‘llanadi. NULL qiymat uchun WHERE ifodasiga o‘xshab quyidagi qoida o‘rinli. Agar izlash sharti NULL qiymatga ega bo‘lsa satrlar guruhi tashlab yuboriladi. HAVING ifodasini GROUP BY siz qo‘llash mumkin. Bu holda natija hamma satrlardan iborat guruh deb qaraladi, lekin amalda bu kam qo‘llanadi.

2.8. FOYDALANUVCHILAR VA ULARNING IMTIYOZLARI

Foydalanuvchilar.

SQL muhitida har bir foydalanuvchi maxsus identifikatsiton nom, murojjat identifikatoriga (ID) ega. Ma’lumotlar bazasiga yuborilgan komanda ma’lum foydalanuvchi bilan yoki boshqacha aytganda maxsus murojaat identifikatori bilan bog‘lanadi. SQL ma’lumotlar bazasida ID ruxsat — bu foydalanuvchi nomi va SQL komanda bilan bog‘langan murojaat identifikatoriga ilova qiluvchi maxsus kalit so‘z USER dan foydalanishi mumkin.

Registratsiya bu kompyuter tizimiga kirish huquqini olish uchun foydalanuvchi bajarishi kerak bo‘lgan protseduradir. Bu protsedura foydalanuvchi bilan qaysi murojaat ID si bog‘lanishini aniqlaydi. Odatda har bir ma’lumotlar bazasidan foydalanuvchi o‘zining ID siga ega bo‘lishi kerak va registratsiya jarayonida haqiqiy foydalanuvchiga aylanadi. Lekin ko‘p masalalarga ega foydalanuvchilar bir necha murojaat ID lari bilan registratsiyadan o‘tishlari, yoki bir necha foydalanuvchi bitta murojaat ID sidan foydalanishlari mumkin.

Imtiyozlar.

Har bir foydalanuvchi SQL ma’lumotlar bazasida nima qilish mumkinlgini ko‘rsatuvchi imtiyozlarga egadir. Bu imtiyozlar vaqt o‘tishi bilan o‘zgarishi, ya’ni eskilari o‘chirilib, yangilari qo‘shilishi mumkin. SQL imtiyozlari bu obyekt imtiyozlaridir. Bu shuni bildiradiki, foydalanuvchi berilgan komandani ma’lumotlar bazasining biror obyekti ustida bajarishi mumkin. Obyekt imtiyozlari bir vaqtning o‘zida foydalanuvchilar va jadvallar bilan bog‘liq. Ya’ni imtiyoz ma’lum foydalanuvchiga ko‘rsatilgan jadvalda, asos jadvalda yoki tasavvurda beriladi. Ixtiyoriy turdagi jadvalni yaratgan foydalanuvchi shu jadval egasidir. Bu shuni bildiradiki, foydalanuvchi bu jadvalda hamma imtiyozlarga ega va imtiyozlarini shu jadvalning boshqa foydalanuvchilariga uzatishi mumkin.

Foydalanuvchiga tayinlash mumkin bo‘lgan imtiyozlar:

- SELECT — Bu imtiyozga ega foydalanuvchi jadvallarda so‘rovlar bajarishi mumkin.
- INSERT — Bu imtiyozga ega foydalanuvchi jadvalda INSERT komandasini bajarishi mumkin.
- UPDATE — Bu imtiyozga ega foydalanuvchi jadvalda UPDATE komandasini bajarishi mumkin. Bu imtiyozni jadvalning ayrim ustunlari uchun cheklab qo‘yishingiz mum-kin.
- DELETE — Bu imtiyozga ega foydalanuvchi jadvalda DELETE komandasini bajarishi mumkin.
- REFERENCES — Bu imtiyozga ega foydalanuvchi jadvalning ustunidan (yoki ustunlaridan) ajdod kalit sifatida foydalanuvchi tashqi kalit aniqlashi mumkin. Siz bu imtiyozni ayrim ustunlar uchun berishingiz mumkin.

Bundan tashqari siz obyekt nostandart imtiyozlarini uchratasiz, masalan INDEX − jadvalda indeks yaratish huquqini beruv-

chi, SYNONYM — obyekt uchun sinonim yaratish huquqini beruvchi va ALTER — jadvalda ALTER TABLE komandasini bajarish huquqini beruvchi. SQL Mexanizm foydalanuvchilarga bu imtiyozlarni GRANT komandasi yordamida beradi.

GRANT Komandasi.

GRANT komandasining 4 formati mavjud bo‘lib, ulardan biri konkret obyekt ustidan konkret foydalanuvchilarga konkret imtyozlar berish bo‘lib, quyidagi ko‘rinishga ega:

GRANT privilege ON [creator.]tablename TO userid, ... [WITH GRANT OPTION]

Bu yerda:

— privilege — tayinlanayotgan imtiyozlar ro‘yxati,

— tablename — jadval nomi,

— userid — imtyozlar olgan foydalanuvchilar ro‘yxati.

Masalan: GRANT SELECT, INSERT ON Orders TO Adrian, Diane

Ma’lum ustunlarga imtiyozlarni cheklanishi.

Bu cheklanish UPDATE va REFERENCES imtiyozlarida ishlatilishi mumkin. Bu holda imtiyoz ko‘rsatilgandan so‘ng qavs ichida shu imtiyoz qo‘llaniluvchi ustunlar ko‘rsatiladi (agar ustunlar ko‘rsatilmagan bo‘lsa, imtiyoz butun jadvalga ta’sir o‘tkazadi).

Masalan:

GRANT UPDATE (City, Comm) ON Salespeople TO Diane; — bu Diane ga Salepeople jadvalining City va Comm ustunlari qiymatlarini o‘zgartirish huquqini beradi yoki GRANT REFERENCES (CName, CNum) ON Customers TO Stephen; — bu komanda Stephen ga CNum va CName ustunlarini o‘zining jadvallaridagi ixtiyoriy tashqi kalitlarga nisbatan ajdod kalit sifatida ishlatish huquqini beradi. Stephen (CName, CNum) yoki (CNum, CName) ustunlarni, jadvalarining ikki ustuni bilan tashqi kalit yordamida mos kelgan ikki ustunli ajdod kalit sifatida aniqlashi mumkin. Yoki u maydonga individual murojaat qilish uchun ajratilgan tashqi kalitlar yaratishi mumkin.

ALL va PUBLIC argumentlaridan foydalanish.

ALL jadvalda hamma imtiyozlarni berish uchun ishlatiladi.

Masalan:

GRANT ALL ON Customers TO Stephen

Agar siz imtiyozlarni publikatsiya (PUBLIC) uchun uzatsangiz, hamma foydalanuvchilar avtomatik ravishda ularni qabul qiladi. Odatda bu ma'lum asos jadvallarda yoki tasavvurlarda (VIEW) foydalanuvchilar imtiyozi uchun qo'llanadi. Ixtiyoriy foydalanuvchiga Buyurtmalar jadvalini ko'rish imkonini berish uchun, siz quyidagini kiritishingiz mumkin:

GRANT SELECT ON Orders TO PUBLIC

WITH GRANT OPTIONS yordamida imtiyozlar berish.

Ba'zida jadval yaratuvchisiga boshqa foydalanuvchilar uning jadvalida imtiyozlarni uzatish imkoniga ega bo'lishlari kerak. Odatda bu bir yoki bir necha xodimlar bir necha yoki hamma asos jadvallarni yaratib, ularni shu jadvallar bilan ishlaydigan xodimlarga topshiradigan tizimlarda zarurdir.

SQL da buning uchun WITH GRANT OPTION ifodasidan foydalaniladi.

Masalan:

Agar Diane buyurtmachilar jadvalida Adrian boshqa foydalanuvchilarga SELECT imtiyozini berish huquqiga ega bo'lishini istasa, unga WITH GRANT OPTION ifodasidan foydalanib SELECT imtiyozini beradi:

GRANT SELECT ON Customers TO Adrian WITH GRANT OPTION

Adrian uchinchi shaxslarga SELECT imtiyozini berish huquqiga ega bo'lgandan so'ng quyidagi komandani berishi mumkin:

GRANT SELECT ON Diane.Customers TO Stephen;

yoki

GRANT SELECT ON Diane.Customers TO Stephen WITH GRANT OPTION

Huquq olgan foydalanuvchilar jadvalga murojaat qilganda jadval egasining murojaat ID sini o'rnatishlari lozim, chunki jadval yaratuvchiga tegishlidir.

Imtiyozlarni rad etish.

Imtiyozlarni REVOKE komandasi yordamida rad etish mumkin, uning sintaksisi GRANT ga o'xshash, lekin teskari ta'sirga ega.

Masalan, Adrian va Stephen uchun sotib oluvchilar jadvalida INSERT va DELETE imtiyozlarini rad etish uchun quyidagi komandadan foydalanish lozim:

REVOKE INSERT, DELETE ON Customers FROM Adrian, Stephen

Imtiyozlarni rad etishda quyidagi qoidalarga rioya qilinadi: imtiyozlar ularni bergan foydalanuvchi tomonidan rad etiladi va rad etish kaskadlanadi, ya'ni undan shu imtiyozlarni olgan hamma foydalanuvchilarga tarqaladi.

Imtiyozlar boshqa turlari (tizim imtiyozlari).

Ma'lumotlar maxsus obyektlari terminlarida aniqlanmaydigan imtiyozlar tizim imtiyozlari yoki ma'lumotlar bazalari qoidalari deb ataladi. Umumiy yondoshishda uchta asosiy tizim imtiyozlari mavjud:

- CONNECT (Ulash),
- RESOURCE (Resurs),
- DBA (Ma'lumotlar Bazasi Administratori).

Soddaroq qilib aytish mumkinki, CONNECT agar obyekt imtiyozlari uzatilgan bo'lsa registratsiya qilinish, tasavvurlar va sinonimlar yaratish huquqidan iborat. RESOURCE asos jadvallarni yaratish huquqidan iborat. DBA bu ma'lumotlar bazasida foydalanuvchiga eng yuqori imkoniyatlar beruvchi superfoydalanuvchi imtiyozidir. Ma'lumotlar bazasi administratori funksiyasiga ega bir yoki bir necha foydalanuvchi shu imtiyozga ega bo'lishi mumkin.

Faqat DBA identifikatorli foydalanuvchi CONNECT, RESOURCE va DBA imtiyozlarini berishi mumkin.

Foydalanuvchiga resurs yoki administrator imtiyozini berish uchun quyidagi komandalarni bajarish yetarli:

GRANT RESOURCE TO userid; yoki mos ravishda
GRANT DBA TO userid.

Foydalanuvchilarni yaratish va o'chirish.

Foydalanuvchini yaratish unga CONNECT imtiyozini uzatish yo'li bilan bajariladi. Komanda sintaksisi quyidagicha:

GRANT CONNECT TO <userid> IDENTIFIED BY <password>

Bu userid nomli foydalanuvchi yaratilishiga olib kelib, unga registratsiya qilinish huquqini beradi va unga parol password tayinlaydi.

Foydalanuvchini o'chirish REVOKE komandasi yordamida CONNECT imtiyozini rad etish orqali amalga oshiriladi. Faqat bazada o'z jadvallariga ega bo'lmagan foydalanuvchini o'chirish

mumkin, chunki bu jadvallar egasiz qoladi. Shuning uchun bunday foydalanuvchini o‘chirishdan oldin uning hamma jadvallarini o‘chirish kerak.

2.9. TARMOQDA MA’LUMOTLAR BAZALARI ARXITEKTURASI

Ma’lumotlar bazasi bilan ishlash amaliy dasturlari arxitekturasi.

Ma’lumotlar bazasi bilan ishlash uchun har xil tillarda amaliy dasturlar yaratish xususiyatlarni ko‘rishdan oldin, bu amaliy dasturlarni loyihalashni ko‘rib chiqish kerak. Biz konseptual tushuncha, ya’ni ma’lumotlar bazasini dasturlash asosida yotadigan klient/ server arxitekturasini ko‘rib chiqamiz. Bu masalalar MySQL va mSQL da dasturlash uchun muhim, lekin faqat ularga xos emas. Aksincha ular ma’lumotlar bazalarini dasturlash ixtiyoriy muhiti uchun muhimdir. Agar arxitektura prinsiplarini hisobga olmasa, sizni amaliy dasturlaringiz talablaringizga javob berolmaydi va o‘zgaruvchi muhitga moslasha olmaydi. Biz quyida murakkab mavzularga, ya’ni oddiy ikki bo‘g‘inli arxitektura, obyektlar va relyatsion ma’lumotlar orasidagi munosabat hamda yangi uch bo‘g‘inli klient/server arxitekturasiga to‘xtalib o‘tamiz.

Klient/server arxitekturasi.

Sodda holda klient/server arxitektura amaliy dasturdagi qayta ishlashni ikki yoki undan ko‘p mantiqiy qismlarga ajratishga asoslangan. Ma’lumotlar bazasi qandaydir amaliy dastur tomonidan foydalanish uchun yaratilgan. Soddalashtirib aytish mumkinki, ma’lumotlar bazasi klient/server arxitekturasining bir qismini tashkil qiladi. Ma’lumotlar bazasi «server», undan foydalanuvchi har qanday amaliy dastur «klient». Odatda klient va server har xil mashinalarda joylashgan; ko‘p hollarda klient amaliy dasturi ma’lumotlar bazasiga do‘stona interfeysdir. Quyidagi grafik shaklda klient/server sodda tizimi tasviri berilgan.

Ma’lumotlar bazasi bilan ishlaydigan amaliy dasturlar yaratilganda avvalambor klientni ma’lumotlar bazasi bilan bog‘lash imkoniyatiga ega bo‘lish kerak. Ma’lumotlar bazalari yaratuvchilari dasturchilardan konkret tilga mo‘ljallangan, API yordamida bog‘lanish asosiy mexanizmlarni berkitadilar. Ma’lumotlar bazasi bilan ishlovchi dastur yaratganingizda sizni

2.1-rasm. Klient/server arxitekturasi.

so‘rovlaringizni tarmoq orqali ma’lumotlar bazasi serveriga uzatiluvchi TCP/IP paketlariga translyatsiya qiladi.

Ma’lumotlar bazasiga murojaat API larining tashqi ko‘rinishi har xil va dasturlash tillariga, ko‘p hollarda ma’lumotlar bazasining o‘ziga bog‘liq. MySQL uchun API lar mSQL bilan o‘xshash qilib yaratilgani uchun, biz ko‘radigan API lar orasidagi farq minimaldir.

Uch bo‘g‘inli arxitektura.

Shu paytgacha biz WWW va biznes amaliy dasturlari bilan ishlash eng sodda arxitekturasi klient/server arxitekturasini muhokama qildik. Lekin bu arxitekturani Amaliy dasturlar rivojlanishi bilan takomillashtirish ancha murakkabdir. Bu arxitekturada obyektga yo‘naltirilgan dasturlash imkoniyatlaridan foydalanish ham qiyin. Birinchi muammo «nozik klientlar» haqidagi bahslarda o‘z aksini topdi. Nozik klientlarga bo‘lgan talab, klientga uzatilayotgan ma’lumotlar o‘sib borish tendensiyasidan kelib chiqdi. Bu muammo PowerBuilder va VisualBasic larda ko‘rindi. Ular bazadan ma’lumotlarni GUI ga oladi va bu ma’lumotlar ustidagi hamma amallarni GUI da bajaradi.

Foydalanuvchi interfeysini baza yadrosiga bog‘lab qo‘yish foydalanuvchilar soni va ma’lumotlar hajmi oshishi bilan o‘zgartirish va masshtablash qiyin bo‘lgan dasturlar yaratilishiga olib keladi. Agar sizda foydalanuvchi interfeysi yaratish tajribasi bo‘lsa, foydalanuvchi xohishiga qarab interfeysni qayta ishlab chiqish muammosiga duch kelgansiz. Bunday qayta ishlashni kamaytirish yo‘li GUI uchun faqat bitta vazifa — foydalanuvchi interfeysi vazifasini qoldirish kerak. Foydalanuvchi bunday interfeysi chindan ham nozik klientdir.

Masshtablanishga ta’sir o‘tkazish boshqa tomondan ham ko‘rinadi. Agar foydalanuvchilar soni va ma’lumotlar hajmi

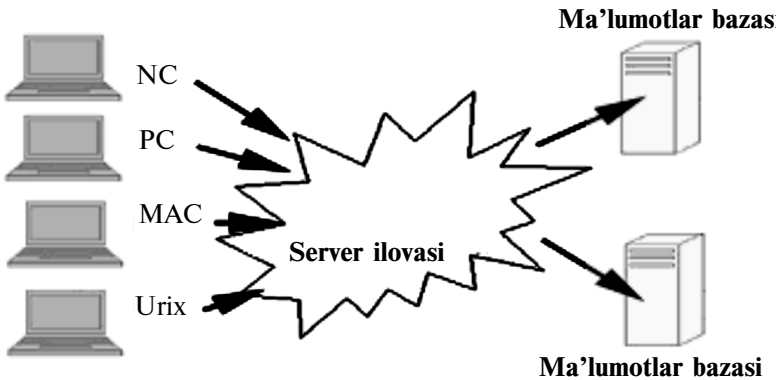
oshgani munosabati bilan amaliy dasturni qayta ishlab chiqish kerak bo‘lsa, modifikatsiya ma’lumotlar bazasiga o‘zgartirish kiritish yo‘li bilan amalga oshirilishi mumkin. Masalan, ma’lumotlar bazasini bir necha serverlarga taqsimlash yo‘li bilan. Interfeysni ma’lumotlar bazasiga bog‘lab qo‘yish masshtablash muammosini hal qilish uchun GUI ni o‘zgartirishga majbur qiladi. Aslida esa bu server bilan bog‘liq muammolardir.

Nozik klientlar — bugunda yagona yo‘nalish emas. Boshqa yo‘nalish — koddan qayta foydalanish. Har xil amaliy dasturlar uchun kod biznes logika deb atalgan qayta ishlashga yo‘naltiriladi. Agar biznes logika foydalanuvchi interfeysida joylashgan bo‘lsa, koddan qayta foydalanishni ta’minlash qiyin bo‘ladi. Bu muammoni hal qilish yo‘li Amaliy dasturni ikki qismga emas uch qismga ajratishdir. Bunday arxitektura uch bo‘g‘inli deyiladi.

Klientdagi foydalanuvchi interfeys haqida gapirganimizda, mantiqiy farqni nazarda tutamiz. Nozik klient bir turi «O‘ta nozik klient» bo‘lib, ko‘pchilik Web-sahifa deb qabul qiladi. Web-sahifa dinamik tarzda Web-serverda yaratilishi mumkin. Bu holda klient ishining ko‘p qismi serverda HTML-sahifalarni dinamik generatsiya qilish shaklida bajariladi.

2.1-rasmda ko‘rsatilgan ikki bo‘g‘inli arxitekturani 2.2-rasmda ko‘rsatilgan uch bo‘g‘inli arxitektura bilan solishtiring. Biz foydalanuvchi interfeysi va ma’lumotlar bazasi orasida qo‘shimcha qatlam joylashtirdik. Bu yangi qatlam amaliy dasturlar serveri o‘zida biror soha uchun umumiy bo‘lgan Amaliy dastur ish mantig‘i — biznes mantiqni oladi. Klient o‘rta yarus obyektlarini ko‘rish vositasi, ma’lumotlar bazasi bo‘lsa shu obyektlar omboriga aylanadi.

Amaliy dasturlar serverining ikki asosiy vazifasi — ma’lumotlar bazasiga ulanishlarni izolyatsiya qilish va biznes mantiq uchun markazlashgan omborni ta’minlash. Foydalanuvchi interfeysi faqat ma’lumotlarni kiritish va akslantirish bilan shug‘ullanadi, ma’lumotlar bazasi yadrosi bo‘lsa faqat ma’lumotlar bazasi muammolari bilan shug‘ullanadi. Ma’lumotlarni qayta ishlashni markazlashtirish Amaliy dasturlar serverining bitta dasturini har xil foydalanuvchi interfeyslari ishlatishi mumkin va har gal yangi amaliy dastur yaratilganda ma’lumotlarni qayta ishlash qoidalarini yozish kerak bo‘lmay qoladi.



2.2-rasm. Uch bo‘g‘inli arxitektura.

2.10. OBYEKTGA YO‘NALTIRILGAN MUROJAAT VA ODBC

Agar siz MySQL uchun yaratilgan dasturni boshqa MBBT ga ko‘chirmoqchi bo‘lsangiz o‘z kodingizni shu yadro API sidan foydalanadigan qilib qayta yozishingiz kerak.

Lekin dasturchilar boshqa ma’lumotlar bazasiga ko‘chirish muammosidan asosan xalos bo‘lganlar. Ularda yagona API, Open DataBase Connectivity API (ODBC), hamma SQL-ma’lumotlar bazalariga unifikatsiya qilingan interfeys mavjud.

ODBC hamma ma’lumotlar bazalariga yagona interfeys bo‘lgani uchun, MySQL va boshqa MBBT lar bilan ishlovchi dasturlar yaratish uchun uni o‘rganib chiqish yetarli. Agar siz kerakli tarzda ODBC dan foydalansangiz, siz yaratgan dasturlar ixtiyoriy MBBT bilan ishlay oladi.

ODBC haqida tushuncha.

Hamma API lar kabi ODBC birgalikda ma’lum funksiyalar to‘plamini ta’minlovchi sinflar va interfeyslar to‘plamidir. ODBC holida bu funksiyalar ma’lumotlar bazasiga murojaatni ta’minlaydi. ODBC API ni tashkil qiluvchi sinflar va interfeyslar ixtiyoriy turdagi ma’lumotlar murojaat qilishdagi umumiy tushunchalar abstraksiyasidir.

Masalan, Connection ma’lumotlar bazasi bilan bog‘lanishni tasvirlovchi interfeysdir. Shunga o‘xshab ResultSet SQL SELECT komandasi qaytaruvchi natijaviy to‘plamni tasvirlaydi. Tabiiyki ma’lumotlar bazasiga murojaat konkret detallari uning yaratuvchisiga bog‘liq. ODBC bu detallar bilan ishlamaydi.

ODBC sinflarini ma'lumotlar bazasini dasturlashga obyektga yo'naltirilgan usullar nuqtai nazaridan ko'rib chiqamiz.

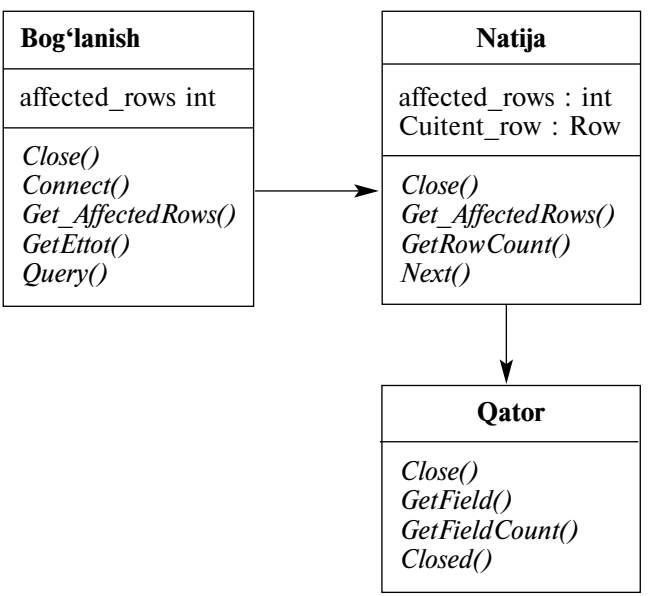
MBBT bilan ishlashni uchta asosiy tushuncha tasvirlaydi: ulanish, natijaviy to'plam va natijaviy to'plam satrlari. 2.3-rasm bu obyektlarni UML-diagrammada ko'rsatadi.

UML — bu yangi Unifikatsiyalangan modellashtirish tili bo'lib, Gradi Buch, Ayvar Yakobson va Djeyms Rambo (Grady Booch, Ivar Jacobson, James Rumbaugh) tomonidan obyektga yo'naltirilgan loyihalash va tahlilni hujjatlash yangi standarti sifatida taklif qilingan.

Ma'lumotlar bazasiga ulanish.

Ixtiyoriy muhitda ma'lumotlar bazasiga murojaat ulanishdan boshlanadi. Bizning obyektga yo'naltirilgan bibliotekamizni yaratish Connection obyektini yaratishdan boshlanadi. Obyekt Connection server bilan bog'lanishni o'rnatish, zarur ma'lumotlar bazasini tanlash, so'rovlarni uzatish va natija olishni bilishi kerak.

Connection sinfi usullari hamma MBBT lar uchun bir xildir. Lekin sinf ichida kompilyatsiya qilinayotgan biblioteka uchun



2.3-rasm. Ma'lumotlar bazasiga obyektga yo'naltirilgan murojaat bibliotekasi.

xos bo‘lgan yopiq a’zolar berkitilgandir. Bog‘lanish o‘rnatishda ma’lumotlar bilan bog‘lanishni ta’minlaydigan sinf a’zodlari farqli bo‘lib qoladi.

Ma’lumotlar bazasi bilan ulanishni o‘rnatish.

Bu API yordamida yaratiladigan hamma amaliy dasturlarga ma’lumotlar bazasiga ulanish uchun Connection sinfi nusxasini uning konstruktorlaridan biri yordamida yaratish kerak bo‘ladi. U kabi uzilish uchun Amaliy dastur Connection nusxasini o‘chirishi kerak. U to‘g‘ridan to‘g‘ri Close() va Connect() usullariga murojaat qilib Connection nusxasini qaytadan ishlatishi mumkin.

Ma’lumotlar bazasidan uzilish.

Connection yana bir mantiqiy funksiyasi ma’lumotlar bazasi bilan aloqani uzish va dasturdan berkitilgan resurslarni ozod qilishdir. Bu funksiyani Close() usuli amalga oshiradi.

Ma’lumotlar bazasiga murojaatlarni bajarish.

Bog‘lanishni ochish va yopishda odatda ma’lumotlar bazasiga komandalar yuboriladi. Connection sinfi argument sifatida SQL komanda oluvchi Query() usuli yordamida bajaradi. Agar komanda so‘rov bo‘lsa 2.3-rasmda ko‘rsatilgan obyekt modelidan Result sinfi nusxasini qaytaradi. Agar komanda ma’lumotlarni yangilayotgan bo‘lsa, usul NULL qaytaradi va affected_rows qiymatini o‘zgartirilgan satrlar soniga teng qiladi.

Natijaviy to‘plamlar.

Result sinfi natijaviy to‘plam ma’lumotlariga hamda shu natijaviy to‘plam bilan bog‘liq metama’lumotlarga murojaatni ta’minlashi kerak. 2.3-rasmda ko‘rsatilgan obyektli modelga asosan bizning Result sinfimiz natijaviy to‘plam satrlarini sikl bo‘yicha o‘qish va undagi satrlar sonini aniqlashni ta’minlaydi.

Natijalar bo‘yicha ko‘chish.

Bizning Result sinfimiz natijaviy to‘plam bilan qatorma-qator ishlaydi. Result sinfi nusxasini Query() usuli yordamida olgandan so‘ng amaliy dastur to navbatdagi Next() usuli 0 qaytarmaguncha, ketma-ket Next() va GetCurrentRow() usullarini chaqirishi lozim.

Satrlar.

Natijaviy to‘plamning alohida satri bizning obyektli modelimizda Row sinfi bilan tasvirlanadi.

Ma'lumotlarga murojaat massiv indeksi bo'yicha so'rov tomonidan berilgan tartibda amalga oshiriladi. Masalan, SELECT user_id, password FROM users so'rov uchun indeks 0 foydalanuvchi nomini va indeks 1 — parolni ko'rsatadi. Bizning C++ API bu indeklashni foydalanuvchi uchun do'stona qiladi. GetField(1) yoki fields[0] birinchi maydonni qaytaradi.

2.11. CGI DAN FOYDALANIB DASTURLASH

CGI haqida tushuncha.

Ko'pgina akronimlar kabi, Common Gateway Interface (CGI — umumiy shlyuzli interfeys) mohiyat haqida hech narsa demaydi. Intefeys nima? Qaerda bu shlyuz? Qanday umuiylik haqida so'z bormoqda? Bu savollarga javob berish uchun oraga qaytamiz va WWW ga nazar tashlaymiz.

Tim Berners-Li, CERN da ishlovchi fizik, Web ni 1990-yilda ishlab chiqdi, lekin reja 1988-yilda yaratildi. G'oya elementar zarralar fizikasi sohasidagi olimlarga Internet orqali tez va yengil multemediya ma'lumotlari — matn, tasvir va tovush bilan almashishdan iborat edi. WWW uchta asosiy qismdan iborat. HTML, URL va HTTP. HTML — Web da saqlanayotgan ma'lumotlarni tasvirlash uchun mo'ljallangan formatlash tili. URL — bu ma'lumotni veb-serverdan HTML formada (yoki boshqa bir) olish uchun ishlatiladigan adre. Va nihoyat, HTTP — bu veb-serverga tanish bo'lgan va klentlarga serverdan hujjatlarni olishga imkon beradigan tildir.

Internet orqali hamma ma'lumotni uzatish imkoniyati revolyutsiya edi, lekin tez orada boshqa imkoniyat aniqlandi. Agar Internet orqali ixtiyoriy ma'lumotni uzatish mumkin bo'lsa, nima uchun tayyor fayldan olinganni emas, dastur tomondan yaratilgan matnni yuborish mumkin emas? Bu bilan juda keng imkoniyatlar yaratiladi. Sodda misol: joriy vaqtni chiqaruvchi dasturdan shunday foydalanish mumkinki, o'quvchi sahifani har bir o'qishda to'g'ri vaqtni ko'radi. Natsional Center for Supercomputing Applications (Superkompyutlar uchun dasturlar yaratish Milliy markazi — NCSA), veb-server yaratish bilan shug'ullanayotgan xodimlari, bu imkoniyatni ko'ra bildilar va tez orada CGI paydo bo'ldi.

CGI — bu serverdagi dasturlar veb-server orqali klentlarga uzatishni belgilovchi qoidalar to'plamidir. CGI Spetsifikatsiya NTML va NTTR larga yangi xarakteristikalar, yangi formalar kiritilishiga olib keldi.

Agar CGI dasturlarga klentlarga ma'lumotlar yuborish bersa, forma klentga dasturlar uchun ma'lumot yuborishga imkon beradi. Endi foydalanuvchi joriy vaqtni ko'rib qolmasdan o'zgartirishi ham mumkin! CGI formalari Web dunyosida chinakam interaktivlik uchun eshik ochdi. Keng tarqalgan CGI Amaliy dasturlar o'z ichiga quyidagilarni oladi.

- Dinamik HNML. Butun bir saytlar bitta CGI-p dastur tomonidan generatsiya qilishi mumkin.
- Foydalanuvchi bergan so'zlar asosida hujjatlarni topishga imkon beruvchi qidiruv mexanizmlari.
- Mehmonlar kitoblari va foydalanuvchilar o'z ma'lumotlarini qoldirishi mumkin bo'lgan e'lonlar taxtalari.
- Byurtmalar blaklari.
- Anketalar.
- Serverda joylashtirilgan ma'lumotlar bazasidan kerakli ma'lumotni ajratib olish.

Keyinchalik biz CGI ma'lumotlar bazalari bilan bog'lanishga imkon beruvchi bu CGI-dasturlarni ko'rib chiqamiz.

CGI spetsifikatsiyasi.

Shunday qilib Illinoys shtati, Bataviyadagi CGI-dasturga Tashqi Mongoliyadagi veb-brouzerga ma'lumotlar almashishga imkon beradigan «qoidalar to'plami» nima o'zi?

CGI to'rtta usul yordamida CGI-darstur va veb-server orasida hamda shunga ko'ra Web klient orasida ma'lumotlar almashadi:

- Atrof muhit o'zgaruvchilari.
- Komanda satri.
- Kiritish standart qurilmasi.
- Chiqarish standart qurilmasi.

Bu to'rtta usul yordamida server klent uzatgan hamma ma'lumotlarni, CGI-dasturga uzatadi. Keyin CGI-dastur o'z ishini bajarib, chiqish ma'lumotlarni serverga, u bo'lsa klentga yubordi.

Atrof-muhit o'zgaruvchilari.

Server CGI-dasturni bajarganda, avvalambor unga atrof-muhit o'zgaruvchilari shaklida ishlash uchun ba'zi ma'lumotlarni yuboradi. Spetsifikatsiyada rasmiy ravishda o'n yettita o'zgaruvchilar ta'riflangan, lekin norasmiy ravishda quyidagi HTTP (nec) zams/n mexanizm yordamida programmu ko'proq ishlatiladi.

CGI-dastur bu o‘zgaruvchilarga, komanda qatorilgan ishga tushirishga komanda protsessorining ixtiyoriy o‘zgaruvchisiga murojaat qilgan kabi murojaat qiladi. Komanda protsessori ssenariyasida, masalan, atrof-muhit o‘zgaruvchisi FOOga *FOO sifatida murojaat qilish mumkin. Bu o‘zgaruvchilar CGI-dastur tomonidan foydalanishi va hatto o‘zgartirilishi mumkin. Lekin bu o‘zgartishlar dasturni ishga tushirgan veb-serverga ta’sir qilmaydi.

Komanda satri.

CGI argumentlari CGI-dasturga komanda qatori satrlari shaklida uzatadi. Bu usul amaliyotda kam qo‘llanadi, shuning uchun biz unga batafsil to‘xtalib o‘tmaymiz. Bu usul mazmuni shundan iboratki, o‘zgaruvchi okrujeniya QUERY STRING ″=″ simvolni o‘z ichiga olmasa, CGI-dastur QUERY STRING dan olingan komanda satri parametrlari bilan bajariladi. Masalan, http: // www. myserver. com/cgi-bin/finger? root satr finger root ni www. myserver. com da ishga tushiradi.

Chiqarish standart qurilmasi.

CGI-dastur tomonidan chiqarish standart qurilmasiga yuboriladigan ma’lumot veb-server tomonidan o‘qiladi va klentga yuboriladi.

Agar ssenariy nomi nph-,dan boshlansa bu ma’lumotlar to‘g‘ri klientga veb-server aralashmasdan yuboriladi. Bu holda CGI-dastur klientga tushunarli bo‘lgan to‘g‘ri HTTP sarlavha hosil qilishi kerak. Aks holda HTTP-sarlavha yaratishni veb-serverga qo‘yib bering.

Agar siz *nph*-ssenariydan foydalanmasangiz, serverga sizni natijangiz haqida ma’lumot beruvchi yana bitta direktiva berish kerak. Odatda bu NTTR-sarlavha Content-Ture, lekin Location sarlavha bo‘lishi ham mumkin. Sarlavhadan so‘ng bo‘sh satr, ya’ni satrni o‘tkazish yoki CR/LF kombinatsiyasi kelishi kerak.

Sarlavha Content-Type serverga, CGI-p dasturingiz qanday turdagi ma’lumot hosil qilgani to‘g‘risida xabar beradi. Agar bu HTML sahifa bo‘lsa, qator quyidagicha bo‘lishi kerak Content-Type: te[y/html. Sarlavha Lokation serverga klentni yuborish kerak bo‘lgan boshqa URL — yoki shu serverdagi boshqa yo‘lni ko‘rsatadi Sarlavha quyidagi ko‘rinishga ega bo‘lishi kerak: http://www. myserver. com/another/place/.

HTTP sarlavhalari va bosh satrdan so‘ng dasturingiz hosil qilgan ma’lumotlarni yuborish mumkin — HTML sahifa, tasvir,

matn yoki yana boshqa ma'lumot. Apache serveri bilan birga o'rnatilishi mumkin bo'lgan CGI-dasturlar ichida *nph-test-cgi* va test-*cgi* mavjud bo'lib, ular nph va nph bo'lmagan usullardagi sarlavhalar orasidagi farqni yaxshi ko'rsatadi.

Holatni eslab qolish.

Holatni eslab qolish faqat jinoyatchilar bilan kurashish uchun emas, foydalanuvchilaringizga yaxshi xizmat qilish uchun ham zarurdir. Muammo kelib chiqish sababi shundaki, HTTP bu «xotirasiz» protokoldir. Bu shuni bildiradiki, klient serverga ma'lumotlar yuboradi, server ma'lumotlarni klientlarga qaytaradi va har biri o'z yo'lida davom etadi. Server klient haqida keyingi amallarda kerak bo'ladigan ma'lumotlarni saqlab qolmaydi. Klient hamma odatda bajarilgan amallar haqidagi ma'lumotni keyinchalik foydalanish uchun saqlab qolmaydi. Bu World Wide Web dan foydalanishga sezilarli chegara qo'yadi.

Bunday protokolda CGI ssenariylarini yaratish, suhbatni eslab qololmaslikka o'xshagandir. Har gal kim bilandir gaplashganda, oldin qancha suhbatlashganingizga qaramasdan, qaytadan tanishishga va suhbat mazmunini izlashga to'g'ri keladi. Netscape Navigator paydo bo'lishi bilan klient qismda cookies deb atalgan shoshilinch yaratilgan usul qo'llana boshladi. B usul klient va server orasida u yoki bu tomonga uzatish mumkn bo'lgan, Content-Type va Location sarlavhalariga o'xshash Yangi HTTP-sarlavha yaratishdan iborat. Klient brouzeri cookie sarlavha olgandan so'ng, cookie da ma'lumotlar va shu cookie ta'sir o'tkazadigan domen nomini saqlab qo'yishi kerak. Shundan so'ng shu domenga tegishli URLga tashrif buyurilganda cookie sarlavha CGI-dasturlarda ishlatilishi uchun serverga qaytarilishi kerak.

Bu usul, ya'ni cookie asosan foydalanuvchi identifikatorini saqlash uchun ishlatiladi. Foydalanuvchi haqidagi ma'lumot server mashinasida saqlanishi mumkin. Bu foydalanuvchining unikal ID si cookie sifatida brouzerga yuborilishi mumkin, shundan so'ng foydalanuvchi har gal saytga tashrif buyurganda, brouzer avtomatik ravishda serverga shu IDni yuboradi. Server IDni CGI dasturga uzatadi, u bo'lsa mos faylni ochadi va foydalanuvchi haqidagi hamma ma'lumotlardan foydalanish imkoniga ega bo'ladi. Bu hammasi foydalanuvchi bilmagan holda yuz beradi.

Bu usul foydaligiga qaramasdan, ko'pgina katta saytlar uni holatni eslab qolishning yagona usuli sifatida ishlatmaydilar.

Buning bir necha sabablari bor. Avvalambor hamma brouzerlar ham cookie ni qo‘llamaydi. Yaqin paytgacha ko‘zi ojiz foydalanuvchilar uchun asosiy brouzer — Lynx —cookie ni qo‘llamas edi.

«Rasmiy» u hali ham qo‘llanmaydi, lekin keng tarqalgan ba’zi «yon shoxlari» qo‘llanadi. Ikkinchidan cookie foydalanuvchini ma’lum mashinaga bog‘lab qo‘yadi. Web ning eng katta ustunligi shundaki, unga dunyoning ixtiyoriy nuqtasidan murojaat qilish mumkin. Sizni veb-sahifangiz qayerda yaratilgani va saqlanayotganidan qat’iye nazar uni ixtiyoriy Internetga ulangan mashinada ko‘rsatish mumkin. Agar siz cookie ni qo‘llovchi saytga begona mashinadan murojaat qilmoqchi bo‘lsangiz, sizni cookie yordamida qo‘llangan shaxsiy ma’lumotlarnigiz yo‘qoladi.

Ko‘pgina saytlar hali ham cookie dan foydalanuvchi sahifalarini personallashtirish uchun foydalanadi, lekin uni an’anaviy «registratsiya nomi/parol» uslubidagi interfeys bilan to‘ldiradi. Agar saytga cookie ni qo‘llamaydigan brouzer orqali murojjat qilinsa, sahifa birinchi kirishda o‘rnatiladigan registratsiya nomi va parol kiritish formasiga ega bo‘ladi. Foydalanuvchilarni cho‘chitmaslik uchun bu forma sodda va kichik bo‘ladi. Foydalanuvchi formaga registratsiya nomi va parol kiritgandan so‘ng, CGI nom xuddi cookie dan yuborilgandek foydalanuvchi haqidagi ma’lumotlarni saqlovchi faylni topadi. Bu usuldan foydalanib foydalanuvchi dunyoning ixtiyoriy nuqtasida personallashgan veb-saytda registratsiyadan o‘tishi mumkin.

CGI va ma’lumotlar bazalari.

Internet davri boshlangandan beri ma’lumotlar bazalari World Wide Web bilan uzviy bog‘langandir. Amaliy jihatdan ko‘pchilik Webni ulkan multimediya ma’lumotlari bazasi deb qaraydi.

Qidiruv mashinalari ma’lumotlar bazalari ustunligini har kuni ko‘rsatadi. Qidiruv mashinasi siz so‘raganda butun Internet bo‘yicha kalit so‘zlarni qidirib o‘tirmaydi. Buning o‘rniga sayt yaratuvchilari boshqa dasturlar yordamida ulkan ko‘rsatkich yaratadilar. Bu ko‘rsatkich qidiruv mexanizmi yozuvlarni ajratib oladigan ma’lumotlar bazasi bo‘lib xizmat qiladi. Ma’lumotlar bazalari ma’lumotni ixtiyoriy murojaatda tez tanlashga imkon beradigan shaklda saqlaydi.

O‘zgaruvchanligi tufayli ma’lumotlar bazalari Web imkonini yanada kengaytiradi: universal interfeysga aylantiradi. Masalan,

tizimli administratorlashni administratorni kerakli tizimda registratsiya qilish talabi o‘rniga masofadan veb-interfeys orqali bajarishga imkon beradi. Ma’lumotlar bazasini Web ga ulash Internet interaktivligi Yangi darajasi asosida yotadi.

Ma’lumotlar bazalarining Web ga ulanish natijasi: dunyodagi axborotning ko‘p qismi ma’lumotlar bazalarida saqlanmoqda.

Web paydo bo‘lmasdan mavjud bo‘lgan ma’lumotlar bazalari merosga olingan (legacy) ma’lumotlar bazalari deyiladi (Oxirgi paytda yaratilgan va Web ga ulanmagan ma’lumotlar bazalariga qarama-qarshi). Ko‘pgina korporatsiyalar (va hatto ayrim shaxslar) bu meros qolgan ma’lumotlar bazalariga Web orqali murojaatni ta’minlashadi. Agar sizni meros qolgan ma’lumotlar bazangiz MySQL yoki mSQL bo‘lmasa, bu mavzu qaral-maydi.

Yuqorida aytilganidek, ma’lumotlar bazalari va Web orasidagi bog‘lanishlar imkoniyatlari cheksizdir. Hozirgi davrda Webdan murojaat qilish imkoniga ega bo‘lgan minglab unikal va foydali ma’lumotlar bazalari mavjud. Bu amaliy dasturlardan tashqari har xil turdagi ma’lumotlar bazalari mavjuddir. Ba’zilari MySQL va mSQL ma’lumotlar bazalari bilan interfeys sifatida CGI-dasturlardan foydalanadilar. Bular biz uchun eng qizi-qarlidir. Boshqalari kommertsial Amaliy dasturlardan keng tarqalgan kichik Microsoft Access i Claris FileMaker Pro kabi ma’lumotlar bazalari bilan bog‘lanish uchun foydalanadilar. Yana boshqalari eng sodda bo‘lgan matnli fayllar bilan ishlaydilar.

Nazorat savollari

1. SQL ilovada yaxlitlikni qanday cheklanish yordamida ta’minlaydi?
2. Kaskadlanuvchi (CASCADE) o‘zgartishlar nima uchun ishlatiladi?
3. Qaysi komandalar COMMIT ni avtomatik bajaradi?
4. Qiymatlar diapazoniga tegishlilikni qanday operator yordamida tekshirish mumkin?
5. NULL AND NULL ifodasi qanday qiymatga teng?
6. Ichki va tashqi jamlashlar orasida qanday farq mavjud?
7. Korrellangan (mutanosib) joylashtirilgan ostki so‘rovlar qanday ishlaydi?
8. EXISTS operatoridan nima uchun foydalaniladi?
9. ALL, ANY, SOME operatorlaridan nima uchun foydalaniladi?

10. COUNT() va COUNT(*) orasida qanday farq mavjud?
11. Agar ustundagi hamma qiymatlar NULL ga teng bo‘lsa, AVG(), SUM(), MIN(), MAX() funksiyalari qanday qiymat qaytaradi?
12. Shart bo‘yicha guruhlarni ajratish uchun qanday operator mavjud?
13. ALL va PUBLIC argumentlaridan nima uchun foydalaniladi?
14. Imtiyozlarni rad etishda qanday qoidalarga rioya qilinadi?
15. Ma’lumotlar bazalari qoidalari deb qanday qoidalarga aytiladi?
16. Qanday identifikatorli foydalanuvchiga CONNECT, RESOURCE va DBA imtiyozlarini berish mumkin?
17. «Nozik klientlar» tushunchasi nimani bildiradi?
18. Connection qanday vazifa bajaradi?
19. Ma’lumotlar bazasiga obyektga yo‘naltirilgan murojaat bibliotekasi qanday sinflardan iborat?

3.1. PHP TILI ASOSLARI

PHP dasturlari.

PHP dasturlari ikki usulda bajarilishi mumkin: Web-server tomonidan ssenariy ilovasi va konsol dasturi sifatida.

Bizning maqsadimiz web-ilovalarni dasturlash bo‘lgani uchun asosan birinchi usulni ko‘ramiz.

PHP odatda Internet bilan bog‘liq dasturlar yaratish uchun ishlatiladi. Lekin PHP dan komanda satrlar interpretatori, asosan *nix tizimlarda foydalanish mumkin. Oxirgisi CORBA va COM interfeyslar hamda PHP-GTK kengaytmasi yordamida mumkin. Bu holda quyidagi masalalarni echish mumkin:

- Interaktiv komanda qatorlari yordamida ilovalar yaratish;
- Kross-platformali GUI ilovalarni PHP-GTK bibliotekasi yordamida yaratish;
- Windows va Linux uchun ba’zi masalalarni avtomatizatsiya qilish.

Serverga brouzerning murojjat qilishi yordamida php-ssenariylari bajarilishini ko‘rib chiqamiz. Avval brouzer .php kengaytmali sahifani so‘raydi, so‘ngra web-server dasturni PHP mashinadan o‘tkazadi va natijani html-kod shaklida qaytaradi. Agar standart HTML sahifani olib, kengaytmasini .php ga o‘zgartirilsa va PHP mashinadan o‘tkazilsa, foydalanuvchiga o‘zgartirmasdan qaytaradi. Bu faylga PHP komandani qo‘shish uchun, PHP komandalarni maxsus teglar ichiga olish kerak. Bu teglarning 4 xil shakli mavjud bo‘lib, ixtiyoriysidan foydalanish mumkin:

1. XML qayta ishlash instruksiyasi:
2. <?php
3. ...
?>
4. SGML qayta ishlash instruktsiyasi:
5. <?
6. ...
?>
7. HTML ssenariylari qayta ishlash instruksiyasi:
8. <script language = "php">

9. ...
10.</script>
11.ASP uslubidagi instruksiya:
12.<%
13. ...
14.%>
Biz XML yoki SGML uslubiga rioya qilamiz.

Izohlar.

PHP tilida izohlarni joylash uchun bir necha usullar mavjud. Eng soddasi ikkilik slesh (//) dan foydalanish, shundan so‘ng PHP satrlar oxirigacha yozilganni o‘tkazib yuboradi. Bundan tashqari (**/*...*/**)ko‘p qatorli izohlardan foydalanish mumkin. Bir qatorli izohlar uchun (#) simvoldan foydalanish qulay. (UNIX script tillaridagi izoh).

```
<php
echo("<p>Hello</p>"); // izoh
echo("<p>Hello</p>"); # izoh
/*
bu ham izoh
*/
?>
```

Shuni esdan chiqarmaslik lozimki, PHP uslubi izohlari faqat PHP chegaralanishlari orasida ta’sir qiladi. Agar PHP bu izohlar simvollarini chegaranishlari tashqarisida uchratsa, ularni boshqa matnga o‘xshab, html- sahifaga joylashtiradi.

Masalan:

```
<php
echo("<p>Hello</p>"); // normal izoh
?>
// bu izoh brouzerda ko‘rinadi.
<!— HTML izohi.
```

Bu izox HTML kodda ko‘rinadi, brouzerda emas.

Izohlarni faqat operator oxiriga emas, quyidagicha joylash ham mumkin:

```
<?
$a = "Hello, world";
echo strstr($a, "H");
// bu funktsiyani keyinchalik qarab chiqamiz
?>
```

O‘zgaruvchilar va konstantalar.

PHP da o‘zgaruvchilar dollar (**$**) belgisidan boshlanadi. Bu simvoldan tashqari ixtiyoriy sondagi harf, raqam va ostiga chiziq simvollari kelishi mumkin, lekin birinchi simvol albatta harf bo‘lishi kerak. Shuni esda tutish kerakki, PHPda o‘zgaruvchilarning nomlari kalit so‘zlardan farqli registrga bog‘liq- dir.

PHP da o‘zgaruvchilarni ta’riflaganda oshkora tipini ko‘rsatish shart emas va dastur davomida bitta o‘zgaruvchi har xil tiplarga ega bo‘lishi mumkin.

O‘zgaruvchi unga qiymat berilganda initsializatsiya qilinadi va dastur bajarilguncha, ya’ni web-sahifa holida to so‘rov tugamaguncha mavjud bo‘ladi.

Tashqi o‘zgaruvchilar.

Klient so‘rovi web-server tomonidan tahlil qilinib, PHP mashinaga uzatilgandan so‘ng, u so‘rovga tegishli ma’lumotlarni o‘z ichiga olgan va bajarish davomida murojaat qilish mumkin bo‘lgan bir necha o‘zgaruvchilarni yaratadi. Oldin PHP sizni tizimingiz atrof-muhit o‘zgaruvchilarini oladi va shu nomdagi va shu qiymatdagi PHP ssenariysi atrofidagi o‘zgaruvchilarni yaratadi, toki serverdagi ssenariylarga klient tizimi xususiyatlari bilan ishlash mumkin bo‘lsin. Bu o‘zgaruvchilar **$HTTP_-ENV_VARS** assotsiativ massivga joylashtiriladi.

Tabiiyki **$HTTP_ENV_VARS** massivi o‘zgaruvchilari tizimga bog‘liqdir (chunki ular aslida atrof-muhit o‘zgaruvchilaridir). Atrof muhit o‘zgaruvchilari qiymatlarini sizni mashinangiz uchun env (Unix) yoki set (Windows) komandasi yordamida ko‘rishingiz mumkin.

So‘ngra PHP GET-o‘zgaruvchilarning guruhini yaratadi. Ular so‘rov satrini tahlil qilishda yaratiladi. So‘rov satri **$QUERY_STRING** o‘zgaruvchida saqlanadi va so‘ralgan URL dagi «**?**» simvoldan keyingi informatsiyadan iborat. PHP so‘rov satrini **&** simvollari bo‘yicha alohida elementlarga ajratadi va har bir elementda «=» belgisini qidiradi. Agar «=» belgisi topilgan bo‘lsa, tenglik chap tomonidagi simvollardan iborat o‘zgaruvchi yaratadi. Quyidagi formani ko‘ramiz:

<form action = "http://localhost/PHP/test.php" method="**get**">
HDD: **<input** type="**text**" name="**HDD**"/>
CDROM: **<input** type="**text**" name="**CDROM**"/>
<input type="**submit**"/>

Agar siz bu formada HDD qatorda ″Maxtor″, CDROM qatorda ″Nec″ tersangiz, quyidagi so‘rov shaklini hosil qiladi:

http://localhost/PHP/test.php?HDD=Maxtor&CDROM=Nec

Bizning misolimizda PHP quyidagi o‘zgaruvchilarni yaratadi: **$HDD** = ″Maxtor″ va **$CDROM** = ″Nec″.

Siz o‘zingizni scriptingizdagi (bizda — test.php) bu o‘zgaruvchilar bilan oddiy o‘zgaruvchilar bilan ishlagandek ishlashingiz mumkin. Bizning misolimizda ular ekranga chiqariladi:

```
<?
echo(″<p>HDD is $HDD</p>″);
echo(″<p>CDROM is $CDROM</p>″);
?>
```

Agar sahifa so‘rovi POST usuli yordamida bajarilsa, POST-o‘zgaruvchilarning guruhi yaratilib, interpretatsiya qilinadi va **$HTTP_POST_VARS** massivga joylashtiriladi.

Konstantalar.

Konstantalar PHP da **define**() funksiyasi yordamida e’lon qilinadi:

define(CONSTANT, value)

Bu funksiya birinchi parametri — konstant nomi, ikkinchisi — uning qiymati. Konstantadan foydalanilganda nomi bo‘yicha ilova qilinadi:

```
<?
define(CONSTANT1,15);
define(CONSTANT2,″\x20″);
define(CONSTANT3,″Hello″);
echo(CONSTANT1);
echo(CONSTANT2);
echo(CONSTANT3);
?>
```

Odatga ko‘ra konstantalar nomlari yuqori registr harflari bilan yoziladi. Konstantalar aniqlanganligini **defined**() funktsiyasi yordamida tekshirish mumkin:

```
<?
define(CONSTANT,″Hello″);
if(defined(″CONSTANT″))
{
echo(″<p>CONSTANT is defined</p>″);
}
?>
```

PHP da ma'lumotlar tiplari.Tiplarni o'zgartirish.

Yuqorida aytilganidek PHP tilida bitta o'zgaruvchini dastur bajarilish davomida satr yoki son sifatida ishlatish mumkin. Shu bilan birga PHP tilida o'zgaruvchilar bilan ishlanganda oshkor ko'rsatilishi musmkin bo'lgan asosiy ma'lumotlar tiplari to'plami mavjud:

- **integer**;
- **string**;
- **boolean**;
- **double**;
- **array**;
- **object**.

PHPo'zgaruvchiga tayinlagan tipni qaytaruvchi **gettype**() funksiyasi mavjud:

```
<?
$var = "5";
$var1 = 5;
echo(gettype($var));
echo "<br>";
echo(gettype($var1));
?>
```

Birinchi holda PHP **string** qaytaradi, ikkinchi holda **integer**.

Tipni oshkora o'rnatuvchi **settype**() funksiyasi mavjuddir:

```
<?
$var = "5";
echo(gettype($var));
settype($var,integer);
echo "<br>";
echo(gettype($var));
?>
```

Kodning bu fragmentini bajarish, avvalgisini bajarish bilan bir xil natijaga olib keladi.

PHP tilida **settype**() funksiyasidan tashqari tipni o'zgartirish operatoridan foydalanish mumkin. Tipni o'zgartirish qavslarga olingan Yangi tipni ko'rsatish orqali bajariladi:

```
$var = (int)$var;
```

PHP quyidagi kodni bajarish natijasida, integer qaytaradi:

```
<?
$var = "5"; // string tip
$var = (int)$var; // int ga o'zgartiramiz
echo(gettype($var));
?>
```

3.2. PHP TILINING OPERATORLARI

Tanlash operatorlari / if...else.

Tanlash operatorlariga: shartli operator **(if...else)** va variantli tanlash operatori **(switch)** kiradi. Shartli operator sintaksisi:

if(condition) statement 1 else statement 2

Shart **condition** ixtiyoriy ifoda bo‘lishi mumkin. Agar u rost bo‘lsa **statement 1** operator bajariladi. Aks holda **statement 2** operatori bajariladi. Shartli operator qisqa shaklida **else** va **statement 2** operatori yozilmaydi.

O‘z o‘rnida **statement 1** va **statement 2** operatorlari shartli operator bo‘lishi mumkin. Bu ixtiyoriy chuqurlikdagi tekshirishlar ketma-ketligini hosil qilishga imkon beradi. Bu ketma-ketlikda shartli operator to‘la yoki qisqa shaklda bo‘lishi mumkin. Shuning uchun **if** va **else** operatorlarini bir biriga mos qo‘yishda xatolik kelib chiqishi mumkin. Tilning sintaksisi bo‘yicha ichki joylashtirilgan shartli operatorlarda har bir **else** eng yaqin **if** ga mos keladi. Xato misol tariqasida quyidagi konstruksiyani keltirish mumkin:

```
<?
$x = 1;
$y = 1;
if($x == 1)
if($y == 1)echo(″x=1 and y=1″);
else echo(″x!=1″);
?>
```

Agar **x** teng **1** va **u** teng **1** bo‘lsa «x = 1 and u = 1» jumla bosmaga chiqariladi. Lekin «x != 1» jumla faqat **x** teng **1** va **u** teng emas **1** holda bosmaga chiqariladi, chunki **else** eng yaqin **if** ga mos keladi. Tashqi shartli operator qisqa shaklda bo‘lib unda **$x == 1** shart tekshiriladi va **statement 1** sifatida **$u == 1** shart tekshiriluvchi to‘la shartli operatorni o‘z ichiga oladi. Ya’ni bu shart faqat **x** teng **1** da bajariladi. Bu masalaning sodda to‘g‘ri echimi figurali qavslardan foydalanib, murakkab shartli operator tuzishdir, Ya’ni figurali qavslar yordamida ichki shartli operatorlarni ajratib, uni qisqa shartli operatorga aylantirishdir. Bu holda tashqi shartli operator to‘la shartli operatorga aylanadi:

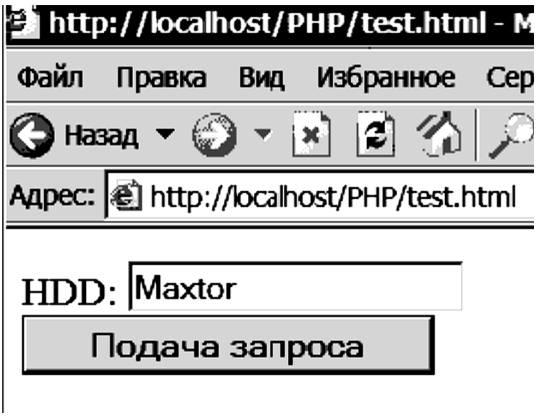
```
<?
$x = 1;
```

```
$y = 1;
if($x==1)
{
if($y==1)echo("x=1 and y=1");
}
else echo("x!=1");
?>
```

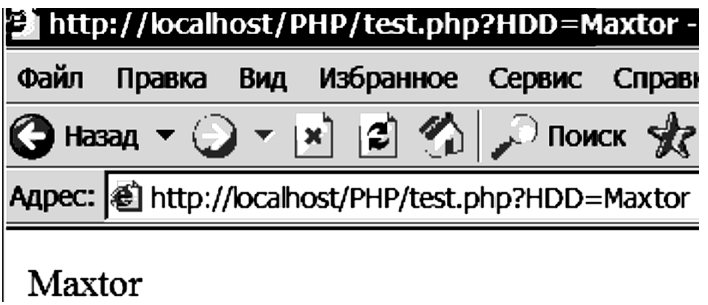
Qo‘shimcha shartlarni **elseif** operatori yordamida tekshirish mumkin. Operator **if** xohlagancha **elseif** bloklarni o‘z ichiga olishi mumkin, lekin **else** har bir **if** operatorida bitta bo‘lishi kerak. Odatda **if...elseif...else** konstruksiyalarda operator **else** boshqa shartlar **true** bo‘lmaganda nima qilish kerakligini aniqlaydi. Umuman elseif operatorining ishlatilishi dastur kodini o‘qishni qiyinlashtiradi, shuning uchun switch dan foydalanish ma’qulroq. PHP shartli operator alternativ sintaksisini ishlatishga imkon beradi. Bu holda shartli operator qavslarsiz yozilib **endif** operatoridan foydalaniladi. Quyidagi misolda agar **$HDD** qiymati «**Maxtor**» ga teng bo‘lsa, ikkinchisining qiymati esa «**Seagate**» ga teng bo‘lsa birinchi jadval sahifaga joylashtiriladi. Bu holda **endif** operator ishlatilishi shart, chunki **if** oxirini ko‘rsatuvchi figurali qavs yo‘q:

```
<?
if($HDD == "Maxtor"):
?>
<table>
<caption> Maxtor </caption>
</table>
<?
elseif($HDD == "Seagate"):
?>
<table>
<caption> Seagate </caption>
</table>
<?
endif;
?>
```

Script bajarilishi natijasi:



So‘rov formasi



Script bajarilishi natijasi $HDD o‘zgaruvchi qiymati test.html forma bilan test.php scriptga uzatiladi. test.html forma kodi:

<**form** action = ″http://localhost/PHP/test.php;″ method=″get″>

HDD: <**input** type=«**text**» name=″**HDD**″/>

<**input** type=″**submit**″/>

</**form**>

PHP tili C++, Java tillari kabi **ifelse** bloklarini shartli amal bilan almashtirishga imkon beradi. Shartli operatsiya (unar va binar amallardan farqli shartli amal uch operand bilan ishlatiladi). Shartli amal yozilishida ketma-ket kelmagan ‘?’ va ‘:’ simvollari hamda uch operand qatnashadi:

ifoda_1 ? ifoda_2 : ifoda_3

Birinchi bo‘lib **ifoda_1** qiymati hisoblanadi. Agar u rost bo‘lsa (ya’ni nolga teng bo‘lmasa) **ifoda_2** hisoblanadi va natijaga aylanadi. Agar **ifoda_1** qiymati nol (yolg‘on) bo‘lsa **ifoda_3** olinadi. Shartli amalga klassik misol quyidagi ifodadir:

x < 0 ? −x : x;

Bu ifoda x o‘zgaruvchi absolyut qiymatini qaytaradi.

Tanlash operatorlari /Variantli tanlash switch.

Variantli tanlash **switch** multitanlash tashkil qilishning eng qulay usulidir. Sintaksisi quyidagicha:

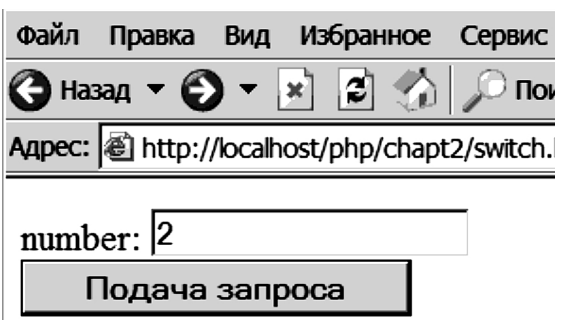
switch(expression) // tanlash ifodasi
{
case value1: // konstantali ifoda 1
statements; // operatorlarning bloki
break;
case value2: // konstantali ifoda 2
statements;
break;
default:
statements;
}

Boshqaruvchi struktura **switch** boshqarishni **case** bilan belgilangan operatorlar ichida, konstantali ifodasi qiymati tanlash ifodasi qiymati bilan teng operatorga uzatadi. Agar tanlash ifodasi qiymati konstantali ifodalarning birortasiga teng bo‘lmasa **default b**ilan belgilangan operatorga o‘tiladi. Har bir variantli tanlash operatorida bittadan ortiq **default** bo‘lishi mumkin emas, lekin u umuman qatnashmasligi ham mumkin. Variantli tanlash operatoridan foydalanilgan dasturga misol keltiramiz. Bu dasturda 1 dan to‘qqizgacha toq raqamlar nomlari chiqariladi, **test.html** formada berilgan songa bog‘liq ravishda. Forma **test.html** oldin foydalanganimizdan farq qilmaydi:

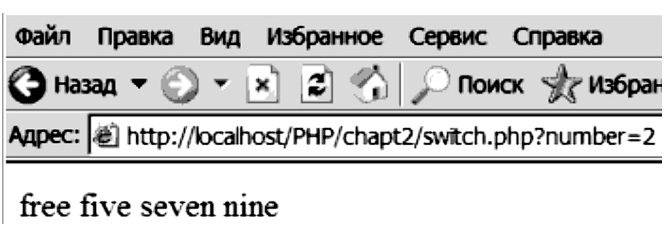
<**form** action = «http://localhost/PHP/chapt2/switch.php» method=″**get**″>
number: <**input** type=″**text**″ name=″**number**″/>
<**input** type=″**submit**″/>
</**form**>
<?
switch($number)
{
case 1:

```
echo ("one");
case 2: case 3:
echo ("free");
case 4: case 5:
echo ("five");
case 6: case 7:
echo ("seven");
case 8: case 9:
echo ("nine");
break;
default:
echo ("This isn't number or number is > 9 or < 1");
}
?>
```

Script bajarilishi natijasi:



So'rov formasi



Scrpt bajarilishi natijasi.

Shartli operatordagi kabi variantli tanlash operatorlari uchun ixtiyoriy darajadagi joylanganlik mumkin, lekin zarur bo'lmasa ko'paytirish kerak emas.

Keltirilgan dasturda **break** operatori ishlatilgan bo‘lib, bu operator variantli tanlash operatoridan chiqishga imkon beradi. Agar **break** operatorlarini har bir raqam chiqarilishidan keyin qo‘yilsa, brouzer oynasida faqat bitta toq son nomini ko‘ramiz.

Ko‘rganimizdek **switch** operatoridan keyin murakkab operator keltirilgan, lekin bu shart emas — **switch** dan so‘ng **case** xizmatchi so‘zi bilan belgilangan ixtiyoriy operator kelishi mumkin.

Eslatamiz: murakkab operator, bu figurali qavsga olingan ixtiyoriy operatorlar ketma-ketligidir.

PHP da **case** belgilari sifatida literallar emas o‘zgaruvchilar ham kelishi mumkin. Lekin **case** belgilari sifatida PHP da massivlar va obyektlar kelolmaydi.

Sikl operatorlari.

Sikl operatorlari sikl tanasida operatorlarning ko‘p marta bajarilishini ta’minlaydi. PHP da 4 ta har xil sikl operatorlari mavjud:

- oldingi shartli sikl:
- **while(condition)**
- **{**
- **statements**;
- **}**
-
- keyingi shartli sikl:
- **do**
- **{**
- **statements**;
- **} while(condition)**;
-
- iteratsion sikl:
- **for(expression1;expression2;expression3)**
- **{**
- **statements**;
- **}**
-
- iteratsion sikl foreach:
- **foreach (array as [$key =>] $value)**
- **{**

- **statements**;
- **}**
-

Agar birinchi sikl operatorlari C-kabi tillardan olingan bo‘lsa, oxirgi operator Perl tilidan olingan (**foreach** siklini biz keyinroq, massivlarni o‘rganganda ko‘ramiz).

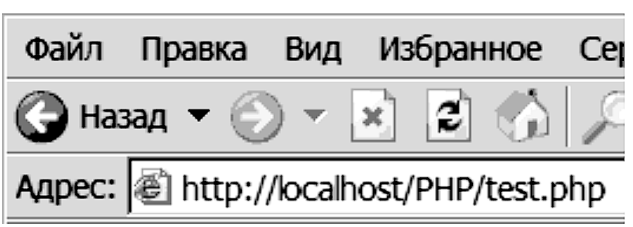
Sikl operatorlari / While.

Operator **while** oldingi shartli sikl operatori deyiladi, siklga kirishda oldin shartli ifoda hisoblanadi, agar uning qiymati noldan farqli bo‘lsa sikl tanasi bajariladi. Shundan so‘ng shartli ifodani hisoblash va sikl tanasi operatorlarini bajarish, shartli ifoda qiymati nolga teng bo‘lguncha davom etadi. **While** operatoridan har xil ketma-ketliklarni ko‘rish qulay, agar ularda oldindan ma’lum so‘nggi simvol mavjud bo‘lsa. (Bu ayniqsa C++ tilida qulay, chunki C++ tilida satr bu char tipidagi nolinchi simvol bilan tugovchi simvollar ketma-ketligidir).

Sodda **while** sikliga misol:

```
<?
$var = 5;
$i = 0;
while(++$i <= $var)
{
echo($i); echo("<br>");
}
?>
```

Bu kod brauzer oynasida birdan beshgacha raqamlarni aks ettiradi:



1
2
3
4
5

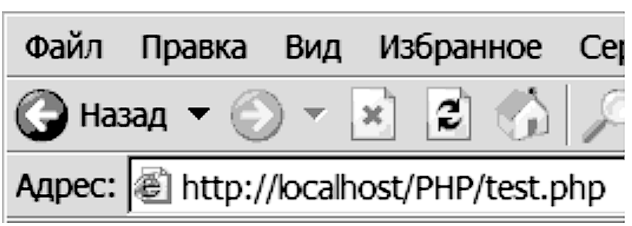
Sikldan chiqish uchun **break** operator qo‘llanadi. Bu operator uchraganda sikl iteratsiyasi to‘xtatiladi. Quyidagi misol bajarilganda, o‘zgaruvchi $var = 7 bo‘lishiga qaramasdan, brouzer oynasida 1 dan 5 gacha raqamlar paydo bo‘ladi.

```
<?
$var = 7;
$i = 0;
while(++$i <= $var)
{
echo($i);
echo("<br>");
if($i==3)break;
}
?>
```

Ba’zida joriy iteratsiyani to‘xtatib, darhol keyingisiga o‘tish kerak bo‘ladi. Buning uchun **continue** operatori qo‘llanadi:

```
<?
$var = 7;
$i = 0;
while(++$i <= $var)
{
if($i==5)
{
continue;
}
echo($i);
echo("<br>");
}
?>
```

Bu misolda 5 raqamidan tashqari 1 dan 7 gacha hamma raqamlar chiqariladi:



1
2

3
4
6
7

Agar siz shartli operatorni **echo** operatoridan keyin qo‘ysangiz, kod xato bo‘ladi va 1 dan 20 gacha raqamlar chiqariladi, chunki ma’lum iteratsiyada sikldan chiqish sharti, shu iteratsiya bajarilgandan so‘ng tekshiriladi.

Cheksiz sikl **while** operatori yordamida quyidagicha hosil qilinadi:

```
while(1)
{
        ...
}
```

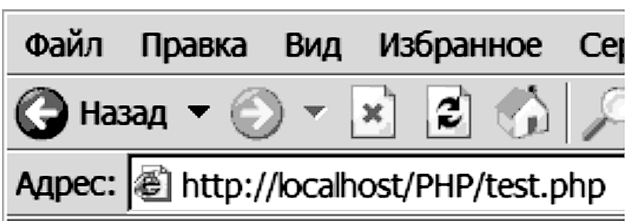
Bu **while**(**true**) yozuvning o‘zi.

Sikl operatorlari / Do…while.

Bu operator keyingi shartli sikl operatori deyiladi. Ixtiyoriy holda siklga kirilganda sikl tanasi bajariladi (ya’ni sikl juda bo‘lmasa bir marta bajariladi) so‘ngra shart hisoblanadi va agar u 0 bo‘lsa yana sikl tanasi bajariladi. Quyidagi misolda nol ro‘yxatga (**++$i <= $var**) shartga bog‘lanmagan holda qo‘shiladi:

```
<?
$var = 5;
$i = 0;
do
{
echo($i); echo("<br>");
}
while(++$i <= $var)
?>
```

Natija:



1
2
3
4
5

Sharti keyin kelgan sikl agar qayta ishlashni tugatish belgisi kelgandan keyin kelganda to‘xtatish zarur bo‘lsa ishlatiladi.

Cheksiz sikl quyidagicha yoziladi:

do ; **while**(1);

Sikl operatorlari/ For

Ta’kidlanganidek iteratsion sikl quyidagi formatga ega:

for(**expression1**;**expression2**;**expression3**)
{
statements;
}

Bu-erda **expression1** (sikl initsializatsiyasi) — vergul bilan ajratilgan ta’riflar va ifodalar ketma-ketligidir. Initsializatsiyaga kirgan hamma ifodalar siklga kirishda bir marta hisoblanadi. Odatda shu yerda sanovchilar boshlang‘ich qiymatlari va sikl parametrlari hisoblanadi. Ifoda — shart (**expression2**) ma’nosi oldingi yoki oxirgi shartli sikllardagi kabi. Agar ifoda — shart bo‘lmasa uning qiymati har doim rost hisoblanadi. Ifoda **expression3** har bir iteratsiya oxirida sikl tanasi bajarilgandan so‘ng hisoblanadi.

Quyidagi scriptda, biz odatga ko‘ra 0 dan 5 gacha sonlarni hisoblaymiz:

```
<?
$var = 5;
$i = 0;
for ($i = 0; $i <= $var; $i++)
{
echo($i);
echo("<br>");
}
?>
```

Natija oldingi rasmda ko‘rsatilganiga o‘xshash.

Cheksizsiklni quyidagicha tashkil qilish mumkin:

for(;;);

yoki

for(;1;)

3.3. MASSIVLAR

Massivlar initsializatsiyasi.

PHP da massivlarni initsializatsiya qilishning 2 usuli mavjud. Birinchisi massiv elementlariga qiymat berishdan iborat:

```
<?
$car[] = "passenger car";
$car[] = "land-rover";
echo($car[1]); // chiqaradi "land-rover"
?>
```

Massiv indeksini ochiq ko‘rsatish mumkin:

```
<?
$car[0] = "passenger car";
$car[1] = "land-rover";
echo($car[1]); // chiqaradi "land-rover"
?>
```

Agar massiv elementlarini e’lon qilishda oshkora indeksatsiyali va indeksatsiyasiz o‘zgaruvchilar aralashib kelsa indeksi berilmagan elementga ishlatilgan indekslar ichida eng kattasidan keyin keluvchi ruxsat berilgan indeksni beradi. Masalan agar biz yaratgan massiv elementlar indekslari 10, 20 va 30 bo‘lsa va indeks ko‘rsatmasdan yangi element yaratsak, uning indeksi avtomatik ravishda 31 bo‘ladi:

```
<?
$car[10] = "passenger car";
$car[20] = "land-rover";
$car[30] = "station-wagon";
$car[] = "victoria";
echo($car[31]);
?>
```

Alternativ usul **array**() konstruksiyasidan foydalanishdan iborat:

```
<?
$car = array("passenger car","land-rover");
echo($car[1]); // chiqaradi "land-rover"
?>
```

Indekslarni oshkora ko‘rsatish uchun => operator qo‘llanadi:

```
<?
$car = array("passenger car", 5 => "land-rover",
"station-wagon","victoria");
```

```
echo($car[0]); echo("<br>"); // chiqaradi "passenger car"
echo($car[5]); echo("<br>"); // chiqaradi "land-rover"
echo($car[6]); echo("<br>"); // chiqaradi "station-wagon"
echo($car[7]); // chiqaradi "victoria"
?>
```

Massiv indekslari satrlar ham bo‘lishi mumkin:

```
<?
$car = array("pc" => "passenger car", "lr" => "land-rover");
echo($car["lr"]); echo("<br>"); // chiqaradi "land-rover"
echo($car["pc"]);  // chiqaradi "passenger car"
?>
```

Massivlarni ko‘rib chiqish uchun foreach sikli.

PHP4 da massiv elementlarini ko‘rib chiqish uchun **foreach** operatoridan foydalanish mumkin. Bu operator sintaksisi:

```
foreach (array as [$key =>] $value)
{
statements;
}
```

Bu sikl ma’nosi sodda: har bir element ko‘rilganda uning indeksi **$key** o‘zgaruvchiga, qiymati bo‘lsa **$value** o‘zgaruvchiga joylashtiriladi. Bu ikki o‘zgaruvchilarning nomlari ixtiyoriydir.

Misol:

```
<?
$car = array("passenger car", "land-rover",
"station-wagon","victoria");
foreach($car as $index => $val)
{
echo("$index -> $val <br>");
}
?>
```

Sintaksisdan ko‘rinib turibdiki, **$key** o‘zgaruvchidan foydalanish shart emas, shuning uchun tashlab yuborilishi mumkin:

```
<?
echo(
"available cars: <br> <ul>"
);
$car = array("passenger car", "land-rover",
"station-wagon","victoria");
foreach($car as $val)
```

```
{
echo("<li>$val</li>\n");
}
echo("</ul>");
?>
```

Ko‘p o‘lchovli massivlar.

Ko‘p o‘lchovli massivlarni ko‘rib chiqish uchun ichki joylashgan **array**() konstruksiyasidan foydalaniladi. Ko‘p o‘lchovli massivlarni o‘qib chiqish joylangan sikllar yordamida amalga oshiriladi. Quyidagi scriptda ko‘p o‘lchovli massiv yaratish va ko‘rib chiqish ko‘rsatilgan.

Misol:

```
<?
$ship = array(
"Passenger ship" => array("Yacht","Liner","Ferry"),
"War ship" => array("Battle-wagon","Submarine",
"Cruiser"),
"Freight ship" => array("Tank vessel","Dry-cargo
ship","Container cargo ship")
);
foreach($ship as $key => $type)
{
echo(
"<h2>$key</h2>\n."<ul>\n");
foreach($type as $ship)
{
echo("\t<li>$ship</li>\n");
}
}
echo(«</ul>\n»);
?>
```

Bu script bajarilish natijasi:

Passenger ship

- Yacht
- Liner
- Ferry

War ship

- Battle-wagon
- Submarine
- Cruiser

Freight ship

- Tank vessel
- Dry-cargo ship
- Container cargo ship

Endi PHP da mavjud massivlar bilan ishlash funksiyalarini ko‘ramiz. Biz massivlarni tartiblash funksiyalardan boshlaymiz. Lekin avval misollarimizda ko‘p foydalanadigan uchta funksiyani ko‘rib chiqamiz.

Funksiya count()

Sintaksis:

int **count(**mixed **var)**

Bu funksiya argument sifatida massivni qabul qilib, undagi elementlar sonini qaytaradi.

Funksiya in_array()

Sintaksis:

boolean **in_array(**mixed **needle**, array **haystack** [, bool **strict**]**)**

Bu funksiya **haystack** massivda **needle** qiymatni qidiradi va agar u mavjud bo‘lsa **true** qaytaradi, aks holda **false** qaytaradi.

Funksiya reset()

Sintaksis:

mixed **reset(**array **array)**

Funksiya **reset**()massiv ko‘rsatkichini birinchi elementga o‘rnatadi va massiv birinchi elementi qiymatini qaytaradi.

Endi tartiblash bilan shug‘ullanamiz.

Massivlarni tartiblash funksiyalari.

sort()

Massivni o‘sish bo‘yicha tartiblash funksiyasi.

Sintaksis:

void **sort(**array **array** [, int **sort_flags**]**)**

Funksiya **array** massivini o‘sish bo‘yicha tartiblaydi. Majburiy bo‘lmagan element **sort_flags** elementlar qanday tartiblanishi kerakligini ko‘rsatadi(tartiblash bayroqlarini belgilaydi). Argumentning mumkin bo‘lgan qiymatlari quyidagilar:

- SORT_REGULAR — elementlarni normal solishtiradi (elementlarni «boricha» solishtiradi);
- SORT_NUMERIC — elementlarni sonlar sifatida solishtiradi;
- SORT_STRING — elementlarni satrlar sifatida almashtiradi.

Umuman olganda bu funksiya ro‘yxatlarni tartiblash uchun mo‘ljallangan. Ro‘yxat deganda kalitlari nuldan boshlangan va bo‘shliklarga ega bo‘lmagan massiv tushuniladi. Funksiya **sort**() ixtiyoriy massivni ro‘yxat deb qaraydi.

Misol:

```
<?
$arr = array("2", "1", "4", "3","5");
sort($arr);
for($i=0; $i < count($arr); $i++)
{
echo ("$i:$arr[$i]");
}
// chiqaradi "0:1 1:2 2:3 3:4 4:5"
?>
```

Natija:

0:1 1:2 2:3 3:4 4:5

Agar siz satrlarni tartilayotgan bo‘lsangiz, misol uchun massiv quyidagi ko‘rinishga ega bo‘lsa array("one", "two", "abs", "three", "uic", "for","five");

bu ajoyib funksiya quyidagi natijani qaytaradi:

Natija:

0:abs 1:five 2:for 3:one 4:three 5:two 6:uic

Ya’ni satrlarni u alfa-beta tartibda, soddaroq aytganda birinchi harflari alifboda kelishi bo‘yicha tartiblaydi.

rsort()

Massivlarni kamayish bo‘yicha tartiblash.

Sintaksis:

void **rsort(**array **arr** [, int **sort_flags**])

Shunga o‘xshash **sort**() funksiyasi faqat kamayish bo‘yicha tartiblaydi. Oldingi **sort**() funksiyasi uchun ko‘rilgan scriptni olamiz, faqat sort(**$arr**) o‘rniga rsort(**$arr**) qo‘yamiz.

Natija:

0:5 1:4 2:3 3:2 4:1

asort()

Assotsiativ massivni o‘sish bo‘yicha tartiblash.

Sintaksis:

void **asort(**array **arr** [, int **sort_flags**])

Funksiya **asort**() berilgan **arr** massivni shunday tartiblaydiki, uning qiymatlari alifbo tartibida (agar satr bo‘lsa) yoki o‘sish tartibda (sonlar uchun) tartibda joylashadi. Bu funksiyaning **sort**() funksiyasidan muhim farqi shundaki,

asort() funksiyasi qo‘llanilganda kalitlar va ularga mos qiymatlar orasida bog‘liqlik saqlanadi, **sort**() funksiyasida bo‘lsa bu bog‘liqlik uziladi.

Misol:

```
<?
$arr = array("a" =>"one","b" => "two","c" => "three","d" => "four");
asort($arr);
foreach($arr as $key => $val)
{
echo ("$key => $val");
}
?>
```

Natija:

d => four a => one c => three b => two

Ko‘rinib turibdiki «kalit-qiymat» bog‘lanishlari saqlanib qolgan.

Ko‘zda tutilgan bo‘yicha **asort**() funksiyasi massivni alifbo bo‘yicha tartiblaydi. Tartiblash bayroqlari **sort_flags** qiymatlari **sort**() funksiyasi ta’rifida keltirilgan.

arsort()

Assotsiativ massivlarni kamayish bo‘yicha tartiblash.

Sintaksis:

void **arsort(**array **arr** [, int **sort_flags**])

Bu funksiya **asort**() funksiyasiga o‘xshash, faqat u massivni o‘sish bo‘yicha emas, kamayish bo‘yicha tartiblaydi.

ksort()

Massivlarni kalit o‘sishi bo‘yicha tartiblash.

Sintaksis:

int **ksort(**array **arr** [, int **sort_flags**])

Bu funksiyada tartiblash qiymatlar bo‘yicha emas, balkim kalitlar bo‘yicha o‘sish tartibida amalga oshiriladi.

```
<?
$arr = array("a" =>"one","b" => "two","c" => "three","d" => "four");
ksort($arr);
foreach($arr as $key => $val)
{
echo ("$key => $val");
}
?>
```

Natija:

a => one b => two c => three d => four

krsort()

Indekslar kamayishi bo‘yicha massivlarni tartiblash.

Sintaksis:

int **krsort(**array **arr** [, int **sort_flags**])

Xuddi **ksort**() funksiyaga o‘xshash, faqat massivni kalitlar bo‘yicha teskari tartibda (kamayish bo‘yicha) tartiblaydi.

array_reverse()

Massiv elementlarini teskari joylashtirish.

Sintaksis:

array **array_reverse(**array **arr** [, bool **preserve_keys**])

Funksiya **array_reverse**()elementlari parametrda berilgan **arr** massivi elementlariga nisbatan teskari joylashtirilgan massivni qaytaradi. Kalitlar va qiymatlar orasidagi bog‘lanish saqlanib qoladi. Agar majburiy bo‘lmagan parametr **preserve_keys** ga **true** berilsa, kalitlar ham teskari tartibda joylashadi.

Misol:

```
<?
$arr = array ("php", 4.0, array ("green", "red"));
$result = array_reverse ($arr);
echo "Massiv: <br>";
foreach($result as $key => $val)
{
echo ("$key => $val <br>");
}
echo("<br>");
echo "Tartiblangan massiv: <br>";
$result_keed = array_reverse ($arr, false);
foreach($result_keed as $key => $val)
{
echo ("$key => $val<br>");
}
?>
```

Birinchi holda:

Natija:

Massiv:
0 =>Array
1 =>4
2 =>php

Tartiblangan massiv:
0 =>Array
1 =>4
2 =>php
Agar ikkinchi parametrga **true** qiymat berilsa:
Natija:
Massiv:
0 =>Array
1 =>4
2 =>php

Tartiblangan massiv:
2 =>Array
1 =>4
0 =>php
shuffle()
Massiv elementlarini tasodifiy joylashtirish.
Sintaksis:
void **shuffle(**array **arr)**
shuffle() funktsiyasi **arr** massivi elementlarini tasodifiy aralashtiradi.
natsort()
Tabiiy tartiblashni bajaradi.
Sintaksis:
void **natsort(**array **arr)**
Bunday tartiblashni satrlarni tartiblashda uchratgan edik. Tabiiy tartiblash deb elementlar tushunarli tartibda joylashga aytiladi.
Misol:
<?
$array1 = $array2 = array**(**"pict10.gif", "pict2.gif", "pict20.gif", "pict1.gif"**);**
echo ("oddiy tartiblash:"**); echo (**"<**br**>"**);**
sort**($array1);**
print_r**($array1);**
echo ("<**br**>"**); echo (**"tabiiy tartiblash:"**); echo (**"<**br**>"**);**
natsort**($array2);**
print_r**($array2);**
?>
Natija:
oddiy tartiblash:

Array ([0] => pict1.gif [1] => pict10.gif [2] => pict2.gif [3] => pict20.gif)

tabiiy saralash

Array ([3] => pict1.gif [1] => pict2.gif [0] => pict10.gif [2] => pict20.gif)

Massiv kursori (ko‘rsatkichi) bilan ishlash.

Yuqorida **reset**() funksiyasi bilan tanishgan edik. Bu funksiya massiv kursorini bo‘shatadi, Ya’ni massiv ichki kursorini massiv boshiga keltiradi va birinchi element qiymatini qaytaradi.

Funksiya **end**() ko‘rilgan **reset**() funksiyaga teskari vazifa bajaradi − kursorni massiv oxiriga keltiradi. Funksiya sintaksis **reset**() funktsiyasi sintaksisiga o‘xshash:

end()

Sintaksis:

mixed **end(**array array arr**)**

next()

Funksiya next() massiv kursorini bitta pozitsiya oldinga suradi.

Sintaksis:

mixed **next(**array array arr**)**

Ya’ni bu funksiya massiv kursorini keyingi elementga o‘rnatadi va siljitishdan oldin kursor joylashgan element qiymatini qaytaradi. Agar massivda qolmagan bo‘lsa **false** qaytaradi. Bu funksiyadan foydalanilganda bir narsani unutmaslik kerak: kursorga bo‘sh element uchragan holda ham **false** qaytariladi. Shuning uchun bo‘sh elementlari bo‘lgan massivlar bilan ishlamoqchi bo‘lsangiz **each**() funksiyasidan foydalangan yaxshiroqdir.

prev()

prev() funksiyasi kursorni bitta pozitsiyaga orqaga suradi. Funksiya Sintaksisi va ishi **next**() funksiyasi bilan bir xil.

Sintaksis:

mixed **prev(**array array arr**)**

current()

Massivning joriy elementini, kursor holatini o‘zgartirmasdan aniqlash uchun **current**() funksiyasi qo‘llanadi.

Sintaksis:

mixed **current(**array array arr**)**

Funksiya **current**() massiv kursori joylashgan elementni qaytaradi va shu bilan birga kursorni siljitmaydi. Agar kursor

massiv tashqarisida bo‘lsa yoki massiv bo‘sh elementlardan iborat bo‘lsa funksiya **false** qaytaradi.

pos()funktsiyasi **current**()funksiyasining to‘la sinonimidir.

key()

Funksiya **key**() massiv joriy elementi indeksini qaytaradi.

Sintaksis:

mixed **key(**array array arr**)**

Endi oldin aytilgan **each**() funksiyasini ko‘ramiz.

each()

Sintaksis:

array **each(**array array arr**)**

Funksiya **each**() massiv joriy elementi «indeks — qiymat» juftligini qaytaradi va massiv kursorini keyingi elementga suradi. Funksiya massiv qaytarib, to‘rtta elementga egadir:

1. [1] => «qiymat»
2. [value] => «qiymat»
3. [0] => indeks
4. [key] => indeks

Agar kursor massiv oxiriga yetgan bo‘lsa, funksiya **false** qaytaradi.

Funksiya ishini ko‘rib chiqamiz:

Misol:

```
<?
$name = array(″maks″, ″eldor″, ″samad″);
$each_name = each($name);
print_r($each_name);
echo(″<br>″);
$each_name = each($name);
print_r($each_name);
echo(″<br>″);
$each_name = each($name);
print_r($each_name);
?>
```

Biz hosil qilgan:

Natija:

Array ([1] => maks [value] => maks [0] => 0 [key] => 0)
Array ([1] => eldor [value] => eldor [0] => 1 [key] => 1)
Array ([1] => samad [value] => samad [0] => 2 [key] => 2)

Funksiya **each**() va funksiya **list**() massiv elementlarini tekshirib chiqish uchun birgalikda qo‘llanishi mumkin. Masalan, quyidagicha:

Misol:

```
<?
$name = array(«maks», «eldor», «samad»);
reset($name);
while(list($key, $val) = each($name))
{
echo («$key = $val<br>»);
}
?>
```

Chiqishdagi:

Natija:

0 = maks
1 = eldor
2 = samad

Funksiya array_walk()

array_walk()

Muhim funksiya bo‘lib, foydalanuvchi funksiyasini massiv har bir elementiga qo‘llashga imkon beradi.

Sintaksis:

bool **array_walk(**array **arr**, callback **func** [, mixed **userdata**])

Bu funksiya sintaksisidan ko‘rinib turibdiki u **func** foydalanuvchi funksiyasini **arr** massivning har bir elementiga qo‘llaydi. Foydalanuvchi funksiyasiga ikki yoki uch argument uzatiladi: joriy element qiymati, indeksi va argument **userdata**. Oxirgi argument majburiy emas. Agar **func** uchtadan ortiq argument talab qilsa va har gal chaqirganda ogohlantirish chiqarilsa, bu ogohlantirish chiqarilmasligi uchun **array_walk**() funksiyasi oldidan «**@**» belgisini qo‘yish kerak. Yana shuni aytish kerakki, **func** funksiyasi **arr** massiv qiymatlari va indekslarini shunday oladiki, ularni o‘zgartira olmaydi. Funksiya qanday qo‘llanishini ko‘rib chiqamiz. Bizga massiv hamma elementlarini chiqarish kerak bo‘lsin. Buning uchun oldin ularni chiqaradigan funksiyani yozib, **array_walk**() funksiyasi yordamida uni chaqiramiz:

Misol:

```
<?
$name = array («m»=>»maks», ″e″=>″eldor″, ″s″=>″samad″);
function print_array ($item, $key)
{
echo ″$key=>$item<br>\n″;
```

```
}
array_walk ($name, "print_array");
?>
```

Nima chiqdi:

Natija:

m=>maksi=>eldors=>samad

Muhim qo‘shimcha. Yuqorida keltirilgan kodda noaniqlik mavjuddir. Ya’ni biz massiv kursorini massiv boshiga o‘rnatmadik va shuning uchun **array_walk**() funksiyasini chaqirishdan oldin shu maqsadda **reset**() funksiyasini chaqirish lozim, chunki **array_walk**() massiv kursori turgan elementdan ishni boshlaydi.

Endi massivni olib, uning har bir elementini birga oshiramiz.

Misol:

```
<?
$number = array ("1"=>"15", "2"=>"20", "3"=>"25");
function printarray ($item, $key)
{
echo "$key=>$item<br>\n";
}
function add_array (&$item, $key)
{
$item = $item + 1;
}
echo("Before:<br>");
array_walk ($number, "printarray");
echo("After:<br>");
array_walk ($number, "add_array");
array_walk ($number, "printarray");
?>
```

Natija:

Before:
1=>15
2=>20
3=>25
After:
1=>16
2=>213=>26

3.4. PHP DA FUNKSIYALAR

Funksiyalarni ta'riflash va chaqirish:

Funksiya **function** kalit so'zi yordamida e'lon qilinadi. Bu kalit so'zdan so'ng figurali qavs ichida funksiya tanasini hosil qiluvchi har xil operatorlar yoziladi:

```
function MyFunction()
{
// operatorlar
}
```

Agar funksiya argumentlar qabul qilsa, ular funksiya ta'rifida o'zgaruvchilar sifatida yoziladi. **Funksiya argumenti** funksiya tanasiga keyingi amallarda qo'llanish uchun uzatiladigan o'zgaruvchidir. Agar funksiya bittadan argumentga ega bo'lsa, bu argumentlar vergul bilan ajratiladi:

```
function MyFunction($var, $var1, $var2)
```

Agar funksiya biror qiymat qaytarsa, funksiya tanasida albatta **return** operatori mavjud bo'lishi kerak:

```
function MyFunction()
{
return $ret; // $ret  o'zgaruvchi qiymati qaytariladi
}
```

Sodda funksiyaga misol.

```
<?
function get_sum()
{
$var = 5;
$var1 = 10;
$sum = $var + $var1;
return $sum;
}
echo(get_sum()); // 15 chiqaradi
?>
```

Bu misolda ikki son summasini hisoblovchi funksiya ko'rsatilgan. Bu funksiya birorta argument qabul qilmaydi, summani hisoblab, natijani chiqaradi. Shundan so'ng **echo** operatori tanasida natijani brauzerga chiqarish uchun chaqiriladi. Bu funksiyani shunday o'zgartiramizki, qiymatni qaytarmasdan, brouzerga chiqarsin. Buning uchun **echo** operatorini funksiya tanasiga kiritish yetarli:

```
<?
```

```
function get_sum()
{
$var = 5;
$var1 = 10;
$sum = $var + $var1;
echo $sum;
}
get_sum();
?>
```

$var va **$var1** o‘zgaruvchilarni argument sifatida e’lon qilishimiz mumkin, bu holda funksiya tangasida ularni tariflash shart emas:

```
<?
function get_sum($var, $var1)
{
$sum = $var + $var1;
echo $sum;
}
get_sum(5,2); // 7 chiqaradi
?>
```

Argument orqali uzatilgan qiymatni o‘z ichiga oluvchi o‘zgaruvchi, funksiya **parametri** deyiladi.

Ko‘rilgan misollarda funksiya argument qiymat bo‘yicha uzatiladi, Ya’ni argumentlar funksiya ichida o‘zgarib, ularning funksiya tashqarisidagi qiymatlariga ta’sir qilmaydi:

```
<?
function get_sum($var)
{
$var = $var + 5;
return $var;
}
$new_var = 20;
echo(get_sum($new_var)); // 25 chiqaradi
echo("<br>$new_var"); // 20 chiqaradi
?>
```

Funksiyaga uzatilgan o‘zgaruvchilar funksiyadan chiqishda qiymatlarini saqlab qolishlari uchun, prametrlarni ilova bo‘yicha uzatish qo‘llanadi. Buning uchun o‘zgaruvchi nomi oldidan ampersand (**&**) belgisi qo‘yiladi:

```
function get_sum($var, $var1, &$var2)
```

Bu holda **$var** va **$var1** o‘zgaruvchilar qiymat bo‘yicha

uzatiladi, **$var2** o‘zgaruvchi bo‘lsa — ilova bo‘yicha. Agar argument ilova bo‘yicha uzatilsa parametr ixtiyoriy o‘zgarishida o‘zgaruvchi — argument ham o‘zgaradi:

```
<?
function get_sum(&$var) // argument ilova bo‘yicha uzatiladi;
{ $var = $var + 5;
return$var;
}
$new_var = 20;
echo(get_sum($new_var)); //25 chiqaradi
echo("<br>$new_var"); // 25 chiqaradi
?>
```

O‘zgaruvchilarning ko‘rinish sohasi.

O‘zgaruvchilar funksiyalarda lokal ko‘rinish sohasiga ega. Bu shuni bildiradiki, hatto lokal va tashqi o‘zgaruvchilar bir xil nomga ega bo‘lsa ham, lokal o‘zgaruvchi o‘zgarishi tashqi o‘zgaruvchiga ta’sir qilmaydi.

```
<?
function get_sum()
{
$var = 5; // lokal o‘zgaruvchi
echo $var;
}
$var = 10; // global o‘zgaruvchi
get_sum(); // 5 chiqaradi  (lokal o‘zgaruvchi)
echo("<br>$var"); // 10 chiqaradi (global o‘zgaruvchi)
?>
```

Lokal o‘zgaruvchini global qilish mumkin, agar uning nomi oldidan **global** kalit so‘zi ko‘rsatilsa. Agar tashqi o‘zgaruvchi **global** sifatida e’lon qilingan bo‘lsa, unga ixtiyoriy funktsiyadan murojaat qilish mumkin:

```
<?
function get_sum()
{
global $var;
$var = 5;
echo $var;
}
$var = 10;
echo("$var<br>");
```

```
get_sum();
?>
```

Global o‘zgaruvchilarga **$GLOBALS** assotsiativ massiv orqali murojaat qilish mumkin:

```
<?
function get_sum()
{
$GLOBALS[«var»] = 20; // global parametrlar o‘zgaradi $var
echo($GLOBALS[″var″]);
}
$var = 10;
echo(″$var<br>″); // 10 chiqaradi
get_sum(); // 20 chiqaradi  (global o‘zgaruvchi o‘zgardi)
?>
```

$GLOBALS massiviga ixtiyoriy funksiya ko‘rinish sohasida murojaat qilish mumkin va u dasturda foydalaniluvchi hamma global o‘zgaruvchilarni o‘z ichiga oladi.

O‘zgaruvchi hayot davri.

O‘zgaruvchi hayot davri deb u mavjud bo‘lgan dastur bajarilish intervali tushuniladi. Lokal o‘zgaruvchilar ko‘rinish sohasi funksiya bo‘lgani uchun, ularning hayot davri ular ta’riflangan funksiya bajarilish vaqti bilan belgilanadi. Bu shuni bildiradiki, har xil funksiyalarda bir biridan mustaqil ravishda bir xil nomli o‘zgaruvchilar ishlatilishi mumkin. Lokal o‘zgaruvchi har gall funksiya chaqirilganda yangidan initsializatsiya qilinadi, shuning uchun quyidagi misolda keltirilgan sanovchi funksiya qaytaruvchi qiymati har gal 1 ga teng bo‘ladi:

```
function counter()
{
$counter = 0;
return ++$counter;
}
```

Lokal o‘zgaruvchi funksiya yangidan chaqirilganda oldingi qiymatini saqlab qolishi uchun uni **static** kalit so‘zi yordamida statik deb e’lon qilish mumkin:

```
function counter()
{
static $counter = 0;
return ++$counter;
}
```

Statik o‘zgaruvchilarning hayot davri ssenariy bajarilish vaqtiga teng. Ya’ni agar foydalanuvchi sahifani qayta yuklasa va natijada ssenariy qaytadan bajarilsa, o‘zgaruvchi **$counter** bu holda yangidan initsializatsiya qilinadi.

Rekursiya tushunchasi.

Rekursiya deb shunday konstruktsiyaga aytiladiki, funksiya o‘zini o‘zi chaqiradi. To‘g‘ri va nisbiy rekursiya ajratiladi. Agar tanasida o‘ziga murojaat bo‘lsa bunday funksiya to‘g‘ri rekursiv deyiladi. Funksiya boshqa funksiyani chaqirsa va bu funksiya o‘z navbatida birinchi funksiyani chaqirsa, bunday funksiya nisbiy rekursiv deyiladi.

Rekursiyani qo‘llashga klassik misollar — darajaga oshirish va son faktorialini hisoblash. Bu misollar rekursiyani tushuntirish qulay bo‘lgani uchun klassik hisoblanadi, lekin ular iteratsion usullarga ko‘ra afzallikka ega emas.

```
<?
function degree($x,$y)
{
if($y)
{
return $x*degree($x,$y-1);
}
return 1;
}
echo(degree(2,4)); // 16 chiqaradi
?>
```

Bu misol quyidagiga asoslangan **xy** ekvivalent **x*x(y-1)**. Bu kodda **24** hisoblash masalasi **2*2к** hisoblashga keltiriladi. So‘ng **2*2к** ni hisoblash **2*2г** ni hisoblashga keltiriladi, toki ko‘rsatkich nolga teng bo‘lmaguncha.

Bu misolning iteratsion varianti quyidagi ko‘rinishga ega:

```
<?
function degree($x,$y)
{   for($result = 1; $y > 0; —$yy)
{
$result *= $x;
}
return $result;
}
echo(degree(2,4)); // 16 chiqaradi
?>
```

Bu kodni tushunish osonligidan tashqari, u samaraliroqdir, chunki siklni bajarish funksiya chaqirishdan arzonga tushadi.

```
<?
function fact($x)
{
if ($x < 0) return 0;
if ($x == 0) return 1;
return $x * fact($x — 1);
}
echo (fact(3)); // 6 chiqaradi
?>
```

3.5. FAYLLAR BILAN ISHLASH

Fayllarni ochish.

Fayl biror ma'lumot saqlash fizik qurilmasidagi baytlar ketma-ketligidir. Har bir fayl joylashuvini ko'rsatuvchi absolyut yo'lga ega. Yo'l ajratuvchisi sifatida Windows da to'g'ri slesh (/) yoki teskari slesh (\) ishlatilishi mumkin. Boshqa operatsion tizimlarda faqat to'g'ri slesh ishlatiladi.

Fayllarni server fayl tizimida ochish **fopen** funksiyasi yordamida amalga oshiriladi:

int fopen(string filename, string mode [, int use_include_path])

Birinchi argument **filename** — fayl nomi yoki unga olib boruvchi absolyut yo'l. Agar absolyut yo'l ko'rsatilmasa, fayl katalogda joylashgan bo'lishi kerak.

Ikkinchi argument **mode** fayl qaysi amallar uchun ochilganligini ko'rsatadi va quyidagi qiymatlarga ega bo'lishi mumkin:

- **r** (faylni faqat o'qish uchun ochish; ochilgandan so'ng fayl ko'rsatkichi fayl boshiga o'rnatiladi);
- **r+** (faylni o'qish va yozish uchun ochish; ochilgandan so'ng fayl ko'rsatkichi fayl boshiga o'rnatiladi);
- **w** (yozish uchun yangi bo'sh fayl yaratish; agar shu nomli fayl mavjud bo'lsa, undagi hamma ma'lumot o'chiriladi);
- **w+** (yozuvlarni o'qish uchun yangi bo'sh fayl yaratish; agar shu nomli fayl mavjud bo'lsa, undagi hamma ma'lumot o'chiriladi);
- **a** (yozuv qo'shish uchun faylni ochish, ma'lumotlar fayl oxiriga yoziladi);

- **a+** (yozuv qo‘shish va o‘qish uchun faylni ochish, ma’lumotlar fayl oxiriga yoziladi);
- **b** (ikkilik fayl bilan ishlash usulini(o‘qishsh va yozish) ko‘rsatuvchi bayroq; faqat Windows da ko‘rsatiladi).

Uchinchi shart bo‘lmagan argument **use_include_path** fayllar **include_path** katalogida izlash kerakligini belgilaydi. (**include_path** parametr php.ini faylda o‘rnatiladi).

Fayl muvaffaqiyatli ochilganda, **fopen** funksiyasi fayl descriptorini qaytaradi, aks holda — **false** qaytaradi. **Fayl Descriptori** ochilgan faylga ko‘rsatkich bo‘lib, opersion tizim tomonidan shu fayl bilan amallarni qo‘llash uchun ishlatiladi. Funksiya tomonidan qaytarilgan fayl descriptorini keyinchalik shu fayl bilan ishlaydigan hamma funksiyalarda ko‘rsatish lozim.

Quyida keltirilgan kod, C:/WWW/HTML/file.txt faylini o‘qish uchun ochadi:

```
<?
$file = fopen("c:/www/html/file.txt","r");
if(!file)
{
echo("Fayl ochish xatoligi");
}
?>
```

Ikkilik fayl, masalan rasmni ochish shu tariqa **b** bayrog‘i binoan bajariladi:

```
<?
$file = fopen("c:/www/html/river.jpg","rb");
if(!file)
{
echo("Fayl ochish xatoligi");
}
?>
```

Fayllarni akslantirish.

Ochilgan fayldagi ma’lumotlarni brouzerda **fpassthru** funksiyasi yordamida akslantirish mumkin:

int fpassthru (int file)

Argument **file** fayl descriptori emasdir.

```
<?
$file = fopen("c:/www/html/pavlovo.jpg","rb");
if(!file)
{
echo("Fayl ochish xatoligi");
```

```
}
else
{
fpassthru($file);
}
?>
```

Matnli fayllarni akslantirish uchun yana bir **readfile** akslantirish funksiyasi mavjuddir:

readfile (string filename)

Shuni ta'kidlash lozimki argument sifatida bu funksiya fayl nomini emas, uning descriptorini qabul qiladi:

```
<?
readfile ("file.txt");
?>
```

Fayllarni berkitish.

Fayl bilan ishni tugallagandan so'ng uni yopish kerak. Fayllarni yopish **fclose** funksiyasi yordamida amalga oshiriladi:

int fclose (int file)

Argument **file** yopish kerak bo'lgan fayl descriptori.

Fayllardan o'qish.

Ochiq fayldan qatorni **fread** funksiyasi yordamida o'qish mumkin:

string fread (int file, int length)

Bu funksiya fayl simvollaridan iborat **length** uzunlikdagi **file** descriptorli qatorni qaytaradi.

```
<?
$file = fopen("c:/www/html/file.txt","r");
if(!file)
{
echo("Fayl ochish xatoligi");
}
else
{
$buff = fread ($file,100);
print $buff;
}
?>
```

Fayldan o'qish uchun **fgets** funksiyasidan foydalanish ham mumkin:

string fgets (int file, int length)

Bu funksiya **length — 1** bayt uzunlikdagi satrni o‘qiydi va qaytaradi. Yangi satr yoki fayl oxiriga yetilganda o‘qish to‘xtatiladi. Fayl oxiriga yetilganda funksiya bo‘sh satr qaytaradi.

HTML teglarini tashlab yuborgan holda faylni o‘qish uchun **fgetss** funksiyasi qo‘llanadi:

string fgetss (int file, int length [, string allowable_tags])

Shart bo‘lmagan uchinchi parametr **allowable_tags** tashlab yuborilmasligi kerak bo‘lgan teglar ro‘yxatidan iborat satrni o‘z ichiga oladi. Bu satrda teglar vergul bilan ajratiladi.

Agar fayldagi bor ma’lumotlarni massivga yozish kerak bo‘lsa, **file** funksiyasi qo‘llanadi:

array file (string filename [, int use_include_path])

Funksiya **filename** nomli faylni o‘qiydi va har bir elementi o‘qilgan fayldagi satrga mos keluvchi massiv qaytaradi. Quyidagi misolda funksiya yordamida fayl o‘qiladi va undagi bor ma’lumot brouzerga chiqariladi.

```
<?
$file_array = file("file.txt");
if(!$file_array)
{
echo("Fayl ochish xatoligi");
}
else
{
for($i=0; $i < count($file_array); $i++)
{
printf("%s<br>", $file_array[$i]);
}
}
?>
```

Bu funksiya qulayligi shundaki, uning yordamida fayldagi satrlar sonini hisoblash mumkin:

```
<?
$file_array =  file ("file.txt");
if(!$file_array)
{
echo("Fayl ochish xatoligi");
}
else
{
$num_str =  count($file_array);
```

```
echo($num_str);
}
?>
```

Shuni e'tiborga olish kerakki, **file** funksiyasini faqat kichkina fayllarni o'qish uchun qo'llash kerak.

Kengaytmasi **.csv* bo'lgan fayllarni o'qish uchun **fgetcsv** funksiyasi qo'llanadi:

array fgetcsv (int file, int length, char delim)

Funksiya fayldan satr o'qiydi va **delim** simvoli bo'yicha uni ajratadi. Parametr delim albatta bir simvoldan iborat satr bo'lishi kerak, aks holda satrlarning faqat birinchi simvoli inobatga olinadi. Funksiya hosil bo'lgan massivni yoki fayl oxiriga yetilgan bo'lsa false qiymatni qaytaradi. Bo'sh satrlar tashlab yuborilmaydi, aksincha bir element bo'sh satrdan iborat massiv qaytaradi. Parametr **length** satrlar maksimal uzunligini qaytaradi, **fgets** funksiyasidagi kabi.

CSV formati MSExcel fayllari saqlanuvchi formatlardan biridir. Quyidagi misolda MSExcel da yaratilgan, foydalanuvchilar parollarini o'z ichiga olgan file.csv o'qiladi.

```
<?
$count = 1;
$file = fopen ("file.csv","r");
while ($data = fgetcsv ($file, 1000, ","))
{
$num = count ($data);
$count++;
for ($i=0; $i < $num; $i++)
{
print "$data[$i]<br>";
}
}
fclose ( $file );
?>
```

Fayllarga yozish.

Fayllarga yozish **fputs** va **fwrite** bir xil funksiyalari bilan amalga oshiriladi:

int fputs (int **file**, **string string [, int length])**

int fwrite (int **file**, **string string [, int length])**

Birinchi argument yozuv amalga oshiralayotgan fayl descriptori. Ikkinchi argument faylga yozilishi kerak bo'lgan satr.

Uchinchi shart bo‘lmagan argument satrda yozilishi kerak bo‘lgan simvollar soni. Uchinchi argument ko‘rsatilmasa hamma satr yozilishi kerak.

Bu misolda ″file.txt″ fayliga ″Hello, world!″ qator yoziladi.

```
<?
$file = fopen (″file.txt″,″r+″);
$str = ″Hello, world!″;
if ( !$file )
{
echo(″Fayl ochish xatoligi″);
}
else
{
fputs ( $file, $str);
}
fclose ($file);
?>
```

Fayllardan nusxa olish, nomini o‘zgartirish va fayllarni o‘chirish.

Fayldan nusxa olish **copy** funksiyasi yordamida amalga oshiriladi:

int copy (string file1, string file2)

copy funksiyasi **file1** nomli fayldan **file2** nomli fayl nusxa oladi. Agar fayl **file2** mavjud bo‘lsa, u qaytadan yoziladi.

Fayl nomini o‘zgartirish **rename** funksiyasi yordamida amalga oshiriladi:

int rename (string old, string new)

Bu funksiya fayl **old** nomini **new** nomiga almashtiradi.

Agar fayl yangi nomi boshqa fayl tizimida joylashgan bo‘lsa **rename** funksiyasi faylga yangi nom bermaydi.

Faylni o‘chirish **unlink** funksiyasi yordamida amalga oshiriladi:

int unlink (string filename)

Fayllar atributlari.

Fayl atributlari haqida qo‘shimcha ma’lumot olish uchun quyidagi funksiyalardan foydalanishingiz mumkin.

file_exists funksiyasi fayl mavjudligini tekshiradi va fayl mavjud bo‘lsa true, aks holda false qiymat qaytaradi:

bool file_exists (string filename)

fileatime funksiyasi faylga oxirgi murojaat vaqtini qaytaradi:

int fileatime (string filename)

filemtime funktsiyasi faylning oxirgi o‘zgartirish vaqtini qaytaradi:

int filemtime (string filename)

file_size funksiyasi fayl hajmini baytlarda qaytaradi:

int file_size (string filename)

file_type funksiyasi fayl tipini qaytaradi:

string file_type (string filename)

Bu funksiya qaytaradigan satr quyidagi fayl tiplaridan biriga tegishli bo‘ladi:

- char (maxsus simvolli qurilma);
- dir (katalog);
- fifo (nomlangan kanal);
- link (simvolli ilova);
- block (maxsus blokli qurilma);
- file (oddiy fayl);
- unknown (tip ma’lum emas).

Fayl xarakteristikalarini qaytaruvchi funksiyalardan foydalanish ko‘p resurslarni talab qilgani uchun, bunday funksiyalarni chaqirishda unumdorlikni yo‘qotmaslik uchun, PHP fayl haqidagi ma’lumotni keshlaydi. Bu keshni **clearstatcache** funksiyasi yordamida tozalash mumkin:

```
<?
clearstatcache();
?>
```

Fayllar bo‘yicha ko‘chish.

Fayldan ma’lumot o‘qilganda joriy pozitsiya ko‘rsatkichi navbatdagi o‘qilmagan simvolga suriladi. Ko‘rsatkich holatini boshqarishga imkon beruvchi bir necha funksiyalar mavjuddir.

Joriy pozitsiya ko‘rsatkichini fayl boshiga keltirish uchun **rewind** funksiyasi qo‘llanadi:

int rewind (int file)

file argumenti fayl descriptoridir.

Ko‘rsatkich joriy pozitsiyasini **ftell** funksiyasi yordamida aniqlash mumkin:

int ftell (int file)

Ko‘rsatkichni fayl ixtiyoriy joyiga **fseek** funksiyasi yordamida o‘rnatish mumkin:

int fseek (int file, int offset [, int whence])

Funksiya **fseek** fayl ko‘rsatkichini **offset** siljishli baytga o‘rnatadi(fayl boshidan, oxiridan yoki joriy pozitsiyadan **whence** parametri qiymatiga qarab). Argument **file** fayl descriptoridir. Argument **whence** qaysi joydan **offset** siljish hisoblanishi kerakligini aniqlaydi va quyidagi qiymatlardan biriga teng bo‘lishi mumkin:

- **SEEK_SET** (pozitsiyani fayl boshidan hisoblaydi);
- **SEEK_CUR** (pozitsiyani ko‘rsatkich joriy pozitsiyasidan hisoblaydi);
- **SEEK_END** (pozitsiyani fayl oxiridan hisoblaydi).

Ko‘zda tutilgan bo‘yicha argument **whence** qiymati SEEK_SET.

Ko‘rsatkich fayl oxirida ekanligini **feof** funksiyasi yordamida aniqlash mumkin:

int feof (int file)

Agar ko‘rsatkich fayl oxirida joylashgan bo‘lsa, funksiya **true** qaytaradi, aks holda false.

feof funksiyasini fayl o‘qishda ishlatish qulaydir:

```
<?
$file = fopen (″file.txt″,″r″);
if ($file)
{
while(!feof($file))
{
$str = fgets($file);
echo $str;
echo (″<br>″);
}
fclose ( $file);
}
else
{
echo(″Fayl ochish xatoligi″);
}
?>
```

Bu funksiya yordamida faylda satrlar sonini aniqlash qulay:

```
<?
$file = fopen (″file.txt″,″r″);
if ($file)
{
$counter = 0;
```

```
while(!feof($file))
{
$str = fgets ($file);
$counter++;
}
echo($counter);
fclose ($file);
}
else
{
echo("Fayl ochish xatoligi");
}
?>
```

Kataloglar bilan ishlash.

Joriy katalogni o‘rnatish uchun **chdir** funksiyasi qo‘llanadi:

int chdir (string directory)

Bu funksiya bilan quyidagicha ishlash mumkin:

- **chdir(**«/tmp/data»**);** // absolyut yo‘l bo‘yicha o‘tish
- **chdir(**«./js»**);** // joriy katalog ost katalogiga o‘tish
- **chdir(**«..»**);** // ajdod katalogga o‘tish
- **chdir(**«~/data»**);** // o‘tamiz /home/foydalanuvchi/data (Unix uchun)

Joriy katalogni aniqlash uchun **getcwd** funksiyasidan foydalanish mumkin:

string getcwd (string path)

Katalogni ochish uchun path parametri bilan berilgan katalogni ochuvchi **opendir** funksiyasidan foydalaniladi:

int opendir (string path)

Katalog ochilgandan so‘ng, uni **readdir** funksiyasi bilan o‘qish mumkin:

string readdir (int dir)

Bu funksiya katalogdagi elementlar nomlarini qaytaradi. Undan tashqari kataloglarda "." va ".." elementlari mavjud. Birinchi element joriy ikkinchisi ajdod katalogni ko‘rsatadi. Joriy katalogning nomini "." sifatida ko‘rsatib ochish mumkin:

$dir = opendir (".");

Katalog bilan ish tugagandan so‘ng uni yopish kerak. Katalog yopilishi **closedir** funksiyasi bilan amalga oshiriladi:

void closedir ($dir)

Quyida joriy katalogdagi fayllarni o‘qish va chiqarishga misol keltirilgan.

```
<?
$dir = opendir (".");
echo "Files:\n";
while ($file = readdir ($dir))
{
echo «$file<br>»;
}
closedir ($dir);
?>
```

Bu funksiya "." va ".." qiymat ham qaytaradi. Agar bu kerak bo‘lmasa, bu qiymatlarni quyidagicha tashlab yuborish mumkin:

```
<?
$dir = opendir (".");
while ( $file = readdir ($dir))
{
if (( $file != ".") && ($file != ".."))
{
echo "$file<br>";
}
}
closedir ($dir);
?>
```

Ko‘rilgan funksiyalarga misol sifatida, c:/temp katalogidagi bir sutka davomida murojaat qilinmagan hamma fayllarni o‘chiruvchi script yaratamiz. Fayllarni o‘chirish funksiyasi bu holda rekursiv chaqiriladi.

```
<?
function delTemporaryFiles ($directory)
{
$dir = opendir ($directory);
while (( $file = readdir ($dir)))
{
if( is_file ($directory."/".$file))
{
$acc_time = fileatime ($directory."/".$file);
 $time =  time();
if (($time — $acc_time) > 24*60*60)
{
if (  unlink ($directory."/".$file))
```

```
{
echo ("Fayllar muvaffaqiyatli o‘chirilgan");
}
}
}
else if ( is_dir ($directory."/".$file) && ($file != ".") &&
($file != ".."))
{
delTemporaryFiles ($directory."/".$file);
}
}
closedir ($dir);
}
delTemporaryFiles ("c:/temp");
?>
```

Kataloglarni yaratish **mkdir** funksiyasi yordamida amalga oshiriladi:

bool mkdir (string dirname, int mode)

Bu funksiya dirname nomli i mode.murojaat huquqlari bilan katalog yaratadi. Katalog yaratilmasa false qaytaradi. Murojaat huquqlari faqat UNIX kataloglari uchun beriladi, chunki Windows da bu argument ta’sir ko‘rsatmaydi. Quyida c:/temp direktoriyasida test v katalogini yaratish misoli ko‘rilgan.

```
<?
$flag = mkdir ("c:/temp/test", 0700);
if($flag)
{
echo("Katalog muvaffaqiyatli yaratilgan");
}
else
{
echo("Katalog yaratish xatosi");
}
?>
```

Katalogni **rmdir** funksiyasi yordamida o‘chirish mumkin:

bool rmdir (string dirname)

Endi yaratilgan /test katalogini o‘chiramiz:

```
<?
$flag = rmdir ("c:/temp/test");
if($flag)
{
```

```
echo(″Katalog muvaffaqiyatli o‘chirilgan″);
}
else
{
echo(″Katalogni o‘chirish xatosi″);
}
?>
```

Funksiya **rmdir** faqat bo‘sh kataloglarni o‘chiradi. Bo‘sh bo‘lmagan kataloglarni o‘chirish uchun, funksiya yaratib c:/temp katalogini undagi papkalar va fayllari bilan birga o‘chiramiz:

```
<?
function full_del_dir ($directory)
{
$dir = opendir($directory);
while(($file = readdir($dir)))
{
if ( is_file ($directory.″/″.$file))
{
unlink ($directory.″/″.$file);
}
else if ( is_dir ($directory.″/″.$file) &&
($file != ″.″) && ($file != ″..″))
{
full_del_dir ($directory.″/″.$file);
}
}
closedir ($dir);
rmdir ($directory);
echo(″Katalog muvaffaqiyatli o‘chirilgan″);
}
full_del_dir (″c:/temp″)
?>
```

Funksiyani rekursiv chaqirganda argumentlar sifatida joriy va ajdod kataloglarga ko‘rsatuvchi ″.″ va ″..″ yozuvlarni uzatmang, chunki bu holda siz ma’lumotlaringizni yo‘qotishingiz mumkin. Bu yozuvlarni shartli operator yordamida o‘tkazib yuboring.

Nazorat savollari

1. PHP dasturlari qanday ikki usulda bajarilishi mumkin?
2. PHPda o‘zgaruvchilarning nomlari registrga bog‘liqmi?
3. Konstantalar PHP da qanday funksiya yordamida e’lon qilinadi?
4. PHP da ma’lumotlar tiplari?
5. Gettype() va settype() funksiyalari qanday vazifani bajaradi?
6. Qo‘shimcha shartlarni qanday operator yordamida tekshirish mumkin?
7. Break va default operatorlari qanday vazifani bajaradi?
8. Sikl operatorlarining qanday turlari mavjud?
9. PHP da massivlarni initsializatsiya qilishning qanday ikki usuli mavjud?
10. Massivlarni ko‘rib chiqish uchun foreach siklidan qanday foydalaniladi?
11. Massivlarni kalit o‘sishi bo‘yicha tartiblash uchun qanday funktsiyalardan foydalaniladi?
12. Prametrlarni ilova va qiymat bo‘yicha uzatishning qanday xususiyatlari mavjud?
13. O‘zgaruvchilar funksiyalarda lokal ko‘rinish sohasiga egaligi nimani bildiradi?
14. Lokal o‘zgaruvchini qanday qilib global qilish mumkin?
15. Lokal o‘zgaruvchi funksiya yangidan chaqirilganda oldingi qiymatini saqlab qolishi uchun uni qanday e’lon qilish kerak?
16. Fayllarni server fayl tizimida ochish qanday funksiya yordamida amalga oshiriladi?

4.1. MySQL SERVERI BILAN ISHLASH

Quyida qanday qilib klient dasturi mysql ga ulanishni ko‘ramiz. Bu dastur yordamida MySQL-serverga ulanish, SQL-so‘rovlarni bajarish va shu so‘rovlar natijalarini ko‘rib chiqish mumkin. Bu qismni o‘rganish uchun kompyuteringizda utilita mysql o‘rnatilgan va MySQL serveri bilan bog‘langan bo‘lishi kerak.

MySQL serveriga mysql dasturi yordamida bog‘lanish uchun foydalanuvchi nomini va odatda parol kiritish lozim. Agar server va klient har xil mashinalarda joylashgan bo‘lsa, MySQL serveri ishga tushirilgan xost nomini ko‘rsatish lozim:

shell> mysql -h host -u user -p

Shundan so‘ng ekranda quyidagi so‘rov paydo bo‘ladi: Enter password:, va sizga o‘z parolingizni kiritishingiz kerak bo‘ladi. Agar ulanish to‘g‘ri amalga oshgan bo‘lsa, ekranda quyidagi ma’lumot va komanda satri belgisi paydo bo‘ladi mysql>:

Welcome to the MySQL monitor. Commands end with; or\g.

Your MySQL connection id is 459 to server version:

Type ‘help’ for help.

mysql>

Quyidagi mysql> belgining paydo bo‘lishi mysql dasturi ishga tayyorligini bildiradi.

Serverdan ixtiyoriy paytda QUIT komandasini terib uzilish mumkin:

mysql> QUIT

Izoh:

Odatda MySQL lokal mashinaga yangi o‘rnatilgan bo‘lsa, murojaat parol va host kiritilmasdan, komanda qatoriga mysql komandasini kiritish yo‘li bilan amalga oshiriladi.

Serverga ulangandan so‘ng komandalar sintaksisini o‘rganish uchun bir necha sodda so‘rovlar berishingiz mumkin. Hali hech qanday ma’lumotlar bazasi tanlanmagani uchun quyida keltirilgan so‘zrovlar umumiy xarakterga ega.

Quyida serverdan versiyasi va vaqtni so‘raydigan sodda komandani keltiramiz:

mysql> SELECT VERSION(), CURRENT_DATE;

MySQL ning bu so‘rovga javobi quyidagi jadvaldan iborat:

```
+---------------+---------------+-------------+
| version() | current_date |
+---------------+---------------+-------------+
1 row in set (0.02 sec)
```

Bu so‘rovni bajarish misolida MySQL bilan ishlash asosiy xususiyatlarini ko‘rish mumkin:

- Serverga yuborilayotgan komanda, odatda SQL-ifodadan iborat bo‘lib, ketidan nuqta vergul keladi. Bu qoidadan chekinishlar bor, masalan QUIT komandasidan so‘ng nuqta vergul qo‘yilmaydi;
- MySQL so‘rov natijasini jadval shaklida chiqaradi;
- So‘rov natijalaridan iborat jadvalni chiqargandan so‘ng, mysql qaytarilgan satrlar soni va so‘rov bajarish vaqtini ko‘rsatadi. Bu qulay, chunki server unumdorligini va so‘rov bajarish effektivligini baholashga imkon beradi;
- So‘rov natijalari va bajarilish vaqtini chiqargandan so‘ng, mysql yangi mysql> satrni chiqaradi, bu esa Yangi komandalar bajarishga tayyorligini ko‘rsatadi.

MySQL komandalari registrga bog‘liq emas, shuning uchun quyidagi so‘rovlar bir xildir:

mysql> select version(), current_date;
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> Select Version(), Current_DATE

MySQL bir satrga bir necha komandalarni joylashtirishga imkon beradi, lekin ular har biri nuqta vergul bilan tugashi kerak. Masalan:

mysql> SELECT VERSION(); SELECT NOW()

Bunday so‘rovga quyidagi natijani olamiz:

```
+--------------+------------+
|  version() |
+--------------+------------+
| 4.0.13-nt |
+--------------+------------+
1 row in set (0.00 sec)
+--------------+------------+
| NOW()                     |
+--------------+------------+
| 2004-01-25 16:57:00 |
+--------------+-------------+
```

1 row in set (0.03 sec)

Lekin hamma komandalarni bir satrga joylash shart emas:

mysql> SELECT USER(),
-> CURRENT_DATE;

Natija:

```
+---------------+---------------+-+---------------+
| user()                    | current_date |
+---------------+---------------+-+---------------+
| ODBC@localhost | 2004-01-25    |
+---------------+---------------+ +---------------+
```

1 row in set (0.00 sec)

E'tibor berinki biz yangi satrga o'tgandan so'ng, komanda satri belgisi mysql> dan -> ga o'zgardi. Bu bilan mysql tugatilgan so'rov olinmaganligini va so'rov oxirini kutayotganligini bildiradi. Bu belgi juda foydali, chunki ba'zi xatolar oldini olishga imkon beradi. Agar siz so'rov oxirida nuqta vergul qo'yishni unutgan bo'lsangiz, mysql bu to'g'rida -> belgini chiqarib bildiradi:

mysql> select user()
->

MySQL dan sodda kalkulyator sifatida foydalanish uchun masalan quyidagi so'rovni kiritish kerak:

mysql> select cos(pi()/10), (2*5)-5;

4.2. MA'LUMOTLAR BAZASIGA MUROJAAT HUQUQINI BERISH

MBBT MySQL o'z ma'lumotlar bazalariga murojaat qilish huquqlarini berish uchun maxsus ma'lumotlar bazasidan foydalanadi. Bu huquqlar serverlar va/yoki foydalanuvchilar nomlariga asoslangan bo'lishlari va bir yoki bir necha ma'lumotlar bazalari uchun berilishlari mumkin.

Foydalanuvchilar akkauntlari parollar bilan ta'minlangan bo'lishi mumkin Ma'lumotlar bazasiga murojaat qilinganda parollar shifrlanadi. Shuning uchun uni o'zgalar bilib olib foydalana olmaydi.

MBBT MySQL uchta jadvalga ega, ya'ni:

Baza: mysql jadvali: db					
Maydon	Tip	Null	Kalit	Belgi	Extra
Host	char(60)		PRI		
Db	char(32)		PRI		
Polzovatel	char(16)		PRI		
Select_priv	char(1)			N	
Insert_priv	char(1)			N	
Update_priv	char(1)			N	
Delete_priv	char(1)			N	
Create_priv	char(1)			N	
Drop_priv	char(1)			N	

Ma'lumotlar bazasi: mysql jadvali: host					
Maydon	Tip	Null	Kalit	Belgi	Extra
Host	char(60)		PRI		
Db	char(32)		PRI		
Select_priv	char(1)			N	
Insert_priv	char(1)			N	
Update_priv	Char(1)			N	
Delete_priv	Char(1)			N	
Create_priv	Char(1)			N	
Drop_priv	Char(1)			N	

Ma'lumotlar bazasi: mysql jadvali: user					
Maydon	Tip	Null	Kalit	Belgi	Extra
Host	char(60)		PRI		
Polzovatel	char(16)		PRI		
Parol	char(8)				
Select_priv	char(1)			N	
Insert_priv	char(1)			N	
Update_priv	char(1)			N	
Delete_priv	char(1)			N	
Create_priv	char(1)			N	
Drop_priv	char(1)			N	
Reload_priv	char(1)			N	
Shutdown_priv	char(1)			N	
Process_priv	char(1)			N	
File_priv	char(1)			N	

Yangi foydalanuvchilar qo‘shishga misol:

$ mysql mysql

mysql> INSERT INTO user VALUES ("%","monty",password("something"),

-> "Y","Y","Y","Y","Y","Y","Y","Y","Y","Y");

mysql> INSERT INTO user (host,user,password) values("localhost","dummy",);

mysql> INSERT INTO user VALUES ("%","admin","N","N","N","N","N","N","Y","N","Y","N");

mysql> quit

$ mysqladmin reload

Uchta yangi foydalanuvchi qo‘shilgan:

monty:	Superfoydalanuvchi (administrator), mysql bilan ishlash uchun paroldan foydalanishi kerak.
dummy:	Individual ma'lumotlar bazasiga "db" jadval bo'yicha murojaat qilishi mumkin.
admin:	Parol kerak emas, lekin faqat "mysqladmin reload" and "mysqladmin processlist" komandalarini bajarishi mumkin. Individual ma'lumotlar bazasiga "db" jadval bo'yicha murojaat qilishi mumkin.

Diqqat! Parolga ega foydalanuvchi yaratish uchun password() funksiyasidan foydalanish kerak. MBBT MySQL shifrlangan parol olishni kutadi.

Foydalanuvchilar jadvalidagi atributlar DB jadvalidagi atributlarni berkitadi. Agar server ko'p ma'lumotlar bazalarini qo'llasa yaxshisi foydalanuvchilar jadvaliga murojaat qilish huquqiga ega bo'lmagan foydalanuvchilarni yaratish va ularga db jadvali bo'yicha ma'lumotlar bazasiga murojaat qilish huquqini berish kerak.

Agar siz MIT threads package dan foydalansangiz, shunga e'tibor beringki, localhost nom ishlamaydi, chunki MIT threads package socket-ulanishni qo'llamaydi. Bu shuni bildiradiki, siz ulanishda har doim hostname (server nomini) aniqlashingiz kerak, bitta server bilan ishlasangiz ham.

Murojaat huquqlarini sozlashda quyidagi qoidalarga amal qilish kerak:

- Server nomi va db jadvaldagi maydon nomi SQL tili regulyar ifodalarini o'z ichiga olishi mumkin: % va _. Boshqa maydonlarda ulardan foydalanish mumkin emas.
- Server nomi domenli nom, localhost nomi, IP adres yoki SQL ifoda bo'lishi mumkin. Bo'sh maydon «server nomi» ixtiyoriy serverni bildiradi.
- Maydon db ma'lumotlar bazasi nomi yoki SQL ifodadir.
- Bo'sh foydalanuvchi nomi ixtiyoriy foydalanuvchiga ekvivalentdir.
- Bo'sh parol ixtiyoriy parolga ekvivalent. Siz superfoydalanuvchi(super-user) yaratishingiz mumkin. Unga foydalanuvchilar jadvali "Y" gida hamma huquqlarni o'rnatish yo'li bilan. Bu foydalanuvchi DB jadvalidagi qiymatlarga qaramasdan ixtiyoriy o'zgartishni qilishi mumkin.

- Serverlar jadvali tekshiriladi faqat va faqat db jadvalida «server nomi» maydoni bo‘sh bo‘lsa.
- Hamma jadvallar host-user-db bo‘yicha tartiblanishi mumkin.

Foydalanuvchi	Serverlar va foydalanuvchilar nomlari bo'yicha tartiblanadi.
Db	Serverlar, foydalanuvchilar va ma'lumotlar bazalari nomlari bo'yicha tartiblanadi.
Server	Serverlar va ma'lumotlar bazalari nomlari bo'yicha tartiblanadi.

Ma’lum foydalanuvchi ma’lum ma’lumotlar bazasiga qanday murojaat qilishini hisoblab chiqish uchun server 3.20.19 versiyasidan boshlab quyidagi ayyorlik kiritilgan. Masalan, quyidagicha registratsiyadan o‘tgan Djo nomli foydalanuvchi mavjud bo‘lsin:

INSERT INTO user VALUES("%.external.domain.com","","","N","N","N", "N","N","N","N","N","N","N")

Ma’lumotlar bazasi jadvalini tekshirish foydalanuvchi Djo uchun emas ‘’ (bo‘sh nom) foydalanuvchi uchun bajariladi, hatto Djo ma’lumotlar bazasiga murojaatlar jadvalida(db jadvalda) yozuvga ega bo‘lsa ham.

Foydalanuvchi akkauntlarini yaratish MBBT MySQL ning eng murakkab ko‘rinuvchi tomonidir. [RTF bookmark end: access]**mysqlaccess** dasturidan foydalanish murojaatni boshqarishni aniqroq qiladi.

Umuman olganda server nomi maydonlarida SQL ifodalardan foydalanmaslik kerak. Bu sozlashni osonlashtiradi.

Aniqroq «server nomi» hamma maydonlarini "%" ga o‘rnating va serverlar jadvalini tozalang. Agar xato olinmasa, serverlar jadvaliga serverlar nomini kitirib ko‘rish mumkin.

Agar "Access denied" xatoligi olinsa, demak siz mysqld demoni bilan to‘g‘ri bog‘langansiz, lekin foydalanuvchilar jadvalida noto‘g‘ri ma’lumotga egasiz.

Parollar qanday ishlaydilar?

Shifrlangan parol foydalanuvchilar jadvalida saqlanadi.

- Ulanish o‘rnatilganda server klientga tasodifiy son yuboradi.

- Klient serverdagi ma'lumotni olish uchun parolni shifrlaydi va serverdn olingan tasodifiy son hamda parol asosida Yangi son hisoblaydi. Bu son serverga yuboriladi.
- Server saqlangan shifrlangan parol va tasodifiy son asosida Yangi son hisoblaydi. Agar bu son klient yuborgan songa teng bo'lsa bog'lanish o'rnatiladi.

4.3. MBBT MySQL DA SQL TILINING REALIZATSIYASI

MySQL ma'lumotlar bazasida ishlatiladigan ma'lumotlar tiplari.

Butun sonlar

Ma'lumotlar tipini ko'rsatish umumiy formasi:

*prefiks*INT [UNSIGNED]

Shart bo'lmagan bayroq UNSIGNED ishorasiz sonlar sonlar saqlash uchun maydon yaratishni bildiradi.

Butun sonlar	
Ma'lumotlar tipini ko'rsatish umumiy formasi:prefiksINT [UNSIGNED] Shart bo'lmagan bayroq UNSIGNED ishorasiz sonlar (o' ga teng yoki katta) sonlar saqlash uchun maydon yaratishni bildiradi.	
TINYINT	Diapazoni — 128 dan 127 gacha sonlarni saqlashi mumkin
SMALLINT	Diapazoni — 32 768 dan 32 767 gacha sonlarni saqlashi mumkin
MEDIUMINT	Diapazoni — 8 388 608 dan 8 388 607 gacha sonlarni saqlashi mumkin
INT	Diapazoni — 2 147 483 648 dan 2 147 483 647 gacha sonlarni saqlashi mumkin
BIGINT	Diapazoni — 9 223 372 036 854 775 808 dan 9 223 372 036 854 775 807 gacha sonlarni saqlashi mumkin

Kasr sonlar

MySQL da butun sonlar bir necha turga ajratilganidek, kasr sonlar ham bir necha turga ajratiladi.Umumiy holda ular quyidagicha yoziladi:Tip nomi[(length, decimals)] [UNSIGNED]

Bu yerda — kasr uzatilishda joylashadigan belgi joylari soni (maydon kengligi).decimals — o‘nli nuqtadan so'ng hisobga olinuvchi raqamlar soni.UNSIGNED — ishorasiz sonlarni beradi.

FLOAT	Aniqligi uncha katta bo'lmagan suzuvchi nuqtali son.
DOUBLE	Ikkilik aniqlikka ega bo'lgan suzuvchi nuqtali son.
REAL	DOUBLE uchun sinonim.
DECIMAL	Satrlar shaklida saqlanuvchi kasr son.
NUMERIC	DECIMAL uchun sinonim.

Satrlar

Satrlar simvollar massivlaridan iborat. Odatda SELECT so‘rovi bo‘yicha matnli maydonlar bo‘yicha izlashda simvollar registri hisobga olinmaydi, Ya’ni "Vasila" va "VASILA" satrlari bir xil hisoblanadi. Agar ma’lumotlar bazasi matni joylashtirish va o’qishda avtomatik qayta kodlashga sozlangan bo’lsa, bu maydonlar siz ko'rsatgan kodlashda saqlanadi. Oldiniga length dan oshmagan simvollar saqlovchi satrlar tiplari bilan tanishamiz, length 1dan to 255 gacha bo‘lgan diapazonda yotadi.

VARCHAR (length) [BINARY]

Bu tipdagi maydonga biror qiymat kiritilganda undan oxirini ko‘rsatuvchi simvollar avtomatik ravishda qirqib olinadi. Agar BINARY bayrog‘i ko‘rsatilgan bo‘lsa, SELECT so‘rovda satr registrni hisobga olgan holda solishtiriladi.

VARCHAR	255 dan ortiq bo‘lmagan simvollarni saqlashi mumkin.
TINYTEXT	255 dan ortiq bo‘lmagan simvollarni saqlashi mumkin.
TEXT	65 535 dan ortiq bo‘lmagan simvollarni saqlashi mumkin.
MEDIUMTEXT	16 777 215 dan ortiq bo‘lmagan simvollarni saqlashi mumkin.
LONGTEXT	4 294 967 295 dan ortiq bo‘lmagan simvollarni saqlashi mumkin.
Ko‘pincha TEXT tpi qo‘llanadi, lekin ma’lumotlar 65 536 simvoldan oshmasligiga ishonmasangiz, LONGTEXT tipidan foydalaning.	
Binar ma’lumotlar	
Binar ma’lumotlar — TEXT formatidagi ma’lumotlarning o‘zi, lekin ularda izlashda simvollar registri hisobga olinadi.	
TINYBLOB	255 dan oshmagan simvollarni saqlashi mumkin.
BLOB	65 535 dan oshmagan simvollarni saqlashi mumkin.
MEDIUMBLOB	16 777 215 dan oshmagan simvollarni saqlashi mumkin.
LONGBLOB	4 294 967 295 dan oshmagan simvollarni saqlashi mumkin.
BLOD-ma’lumotlar avtomatik qayta kodlanmaydi, agar o‘rnatilgan ulanish bilan ishlaganda darhol qayta kodlash imkoniyati o‘rnatilgan bo‘lsa.	
Sana va vaqt	
MySQL sana va vaqtni har xil formatlarda saqlash uchun mo‘ljallangan maydonlar bir necha tiplarini qo‘llaydi.	

DATE	GGGG-MM-DD formatdagi sana
TIME	CHCH:MM:SS formatdagi vaqt
DATETIME	GGGG-MM-DD CHCH:MM:SS formatdagi sana va vaqt timestamp formatdagi sana va vaqt.
TIMESTAMP	Lekin maydon qiymatini olishda u timestamp formatida emas, GGGGMMDDCHCHMMSS formatda aks etadi, bu esa PHP dan undan foydalanish qimmatini ancha kamaytiradi.

MySQL ma'lumotlar bazasini yaratish (CREATE DATABASE).

Ma'lumotlar bazasi CREATE DATABASE komandasi yordamida yaratiladi.

Komanda sintaksisi:

CREATE DATABASE *database_name*

□ *database_name* — Ma'lumotlar bazasiga beriladigan nom.

Keyingi misolda db_test ma'lumotlar bazasini yaratamiz:

CREATE DATABASE db_test

PHP da ma'lumotlar bazasini yaratish:

```
$sql="CREATE DATABASE db_test";
mysql_query($sql)
```

MySQL ma'lumotlar bazasini o'chirish (DROP DATABASE).

Ma'lumotlar bazasini o'chirish uchun **DROP DATABASE** komandasidan foydalaniladi.

Sintaksis:

DROP DATABASE *database_name*

Bu yerda:

□ *database_name* — o'chirish kerak bo'lgan ma'lumotlar bazasi nomi.

Quyidagi misolda db_test ma'lumotlar bazasi o'chiriladi:

DROP DATABASE db_test

PHPda ma'lumotlar bazasini o'chirish:

```
$sql="DROP DATABASE db_test";
mysql_query($sql)
```

USE.

Jadvallar bilan ishlash uchun MySQL ga qaysi baza bilan

ishlash haqida ma'lumot berish kerak. Bu USE komandasi yordamida amalga oshiriladi:

USE db_name;

Bu yerda db_name — tanlangan ma'lumotlar bazasi nomi. Yaratilgan db_test bazasini tanlaymiz:

mysql> CREATE DATABASE db_test;
Database changed

MySQL ma'lumotlar bazasida jadval yaratish (CREATE TABLE).

Jadval yaratish **CREATE TABLE k**omandasi orqali amalga oshiriladi.

CREATE TABLE table_name(column_name1 *type*, column_name2 *type*,...)

- table_name — Yangi jadval nomi;
- column_name — yaratilayotgan jadval ustunlari (maydonlari), nomlari.
- *type* — ustun tipi.

Do'stlaringiz telefon nomerlari jadvalini yaratish kerak bo'lsin.

Bizning jadvalimiz uch ustundan iborat bo'ladi: Do'stingiz ismi sharifi, adres va telefoni

CREATE TABLE tel_numb(fio text, address text, tel text)

PHP da bu quyidagi ko'rinishga ega bo'ladi:

$sql="CREATE TABLE tel_numb(fio text, address text, tel text)";
mysql_query($sql)

Ma'lumotlar turlariga mos ustunlar bilan bajarish mumkin bo'lgan (yoki taqiqlangan) operatsiyalarni ko'rsatuvchi modifikatorlarni ulash mumkin.

not null — Maydon noma'lum qiymatga ega bo'la olmasligini, ya'ni jadvalga yangi yozuv qo'shishda maydon albatta initsializatsiya qilinishi kerakligini (agar ko'zda tutilgan qiymat berilmagan bo'lsa) ko'rsatadi.

Masalan, bizning telefonlar jadvalimizda do'stimiz ismi sharifi (maydon fio) va telefoni (maydon tel) maydonlari noma'lum qiymatga ega bo'la olmasligini ko'rsatish kerak:

CREATE TABLE tel_numb(fio text NOT NULL, address text, tel text NOT NULL)

primary key — Maydon birlamchi kalitligini, ya'ni ilova qilish mumkin bo'lgan yozuv identifikatori ekanligini aks ettiradi.

CREATE TABLE tel_numb(fio text, address text, tel text, PRIMARY KEY (fio))

auto_increment — Maydonga yangi yozuv qo‘shishda maydon unikal qiymat qabul qiladi va jadvalda hech qachon bir xil nomerli maydonlar mavjud bo‘lmaydi.

CREATE TABLE tel_numb(fio text AUTO_INCREMENT, address text, tel text)

default — maydon uchun ko‘zda tutilgan qiymatni aniqlaydi. Agar joylanayotgan yozuvda bu maydon uchun qiymat ko‘rsatilmagan bo‘lsa, shu qiymat kiritiladi.

CREATE TABLE tel_numb(fio text, address text DEFAULT ″Ko‘rsatilmagan″, tel text)

SHOW Komandalar.

Ma’lumotlar bazasi muvaffaqiyatli yaratiganini tekshirish uchun kompyuteringizda qanday ma’lumotlar bazasi mavjudligini ko‘rsatuvchi SHOW DATABASES komandasini bajarish lozim:

mysql> SHOW DATABASES

Hamma jadvallar muvaffaqiyatli yaratilganiga ishonch hosil qilish uchun, SHOW TABLES komandasini bajaramiz.

Tanlangan jadval hamma ustunlari ro‘yxatini quyidagi so‘rov yordamida chiqarish mumkin:

mysql> SHOW FIELDS FROM tel_numb

DESCRIBE.

DESCRIBE komandasi yaratilgan jadvallar strukturasini ko‘rsatadi va quyidagi sintaksisga ega:

DESCRIBE table_name

Bu yerda table_name — strukturasi so‘ralayotgan jadval.

DESCRIBE Komanda SQL standartiga kirmaydi va MySQL ichki komandasidir.

Keling, quyidagi SQL- so‘rov bajarib forums jadvali strukturasini ko‘ramiz:

mysql> DESCRIBE tel_numb

MySQL ma’lumotlar bazasidan jadvalni o‘chirish (DROP TABLE).

Jadvalni **o‘chirish** uchun **DROP TABLE** komandasidan foydalaniladi:

DROP TABLE *table_name*

□ *table_name* — o‘chirilayotgan jadval nomi.

DROP TABLE tel_numb

PHP da bu quyidagi ko‘rinishga ega bo‘ladi:

$sql=″DROP TABLE tel_numb″
mysql_query($sql)

Jadval xossalarini o‘zgartirish: Jadvalni qayta nomlash (ALTER TABLE RENAME).

Jadvalga yangi nom berish quyidagi konstruktsiya yordamida amalga oshirilishi mumkin:

ALTER TABLE *table_name_old* **RENAME** *table_name_new*

Bu yerda:

□ *table_name_old* — jadval eski nomi;
□ *table_name_new* — jadval yangi nomi.

Misol uchun *search* jadvali nomini *search_en* nomiga o‘zgartirish kerak bo‘lsin:

$sql=″ALTER TABLE search RENAME search_en″;
mysql_query($sql)

Jadvallar xossalarini o‘zgartirish:Ustunlar qo‘shish (ALTER TABLE ADD).

Yangi ustun qo‘shishni quyidagi konstruktsiya yordamida amalga oshirish mumkin:

ALTER TABLE *table_name* **ADD** *field_name* parametrs.

Bu yerda:

□ *table_name* — yangi ustun qo‘shiladigan jadval nomi;
□ *field_name* — qo‘shilayotgan ustun nomi;

parametrs — qo‘ilayotgan ustunni tasvirlovchi parametrlar. Ma’lumotlar tipini ko‘rsatish majburiy parametrdir.

Masalan, *my_frends* nomli jadvalga *adress_2* nomli matn qiymatlarga ega ustun qo‘shishimiz kerak bo‘lsin:

$sql=″ALTER TABLE my_frends ADD adress_2 TEXT″;
mysql_query($sql)

Ko‘zda tutilgan bo‘yicha yangi ustun jadval oxiriga qo‘shiladi.

Agar ustun jadval boshiga qo‘shilishi kerak bo‘lsa, qo‘shilayotgan ustun parametrlaridan so‘ng **FIRST** kalit so‘zini yozish kerak:

$sql=″ALTER TABLE my_frends ADD adress_2 TEXT FIRST″;
mysql_query($sql)

Agar ustun jadval boshi yoki oxiri emas, balki ma'lum ustundan keyin qo'yilishi lozim bo'lsa quyidagi kalit so'zdan foydalanish lozim: **AFTER** *ustun nomi, shu ustundan so'ng yangi ustun qo'shiladi:*

$sql="ALTER TABLE my_frends ADD adress_2 TEXT AFTER adress_1";

mysql_query($sql)

Bu misolda yangi *adress_2* ustuni *adress_1* ustunidan keyin qo'yiladi.

Agar jadvalga bir emas bir necha ustun qo'shish kerak bo'lsa har bir ustun uchun **ADD** *field_name* parametrs vergul orqali yozish kerak:

$sql="ALTER TABLE my_frends ADD adress_2 TEXT,
ADD adress_3 TEXT, ADD adress_4 TEXT";
mysql_query($sql);

Agar jadvalga ikki ustun qo'shish lozim bo'lsa, quyidagicha amalga oshirish mumkin:

$sql="ALTER TABLE my_frends ADD adress_2 TEXT AFTER adress_1,
ADD adress_3 TEXT AFTER adress_2";
mysql_query($sql);

Ya'ni birinchi qo'shilayotgan ustunni *adress_1* dan so'ng, ikkinchisini birinchisidan so'ng.

Jadval xossalarini o'zgartirish: Ustun xossalarini o'zgartirish (ALTER TABLE CHANGE).

Bir yoki bir necha ustunlar xossalarini quyidagi konstruktsiya yordamida o'zgartirish mumkin:

ALTER TABLE *table_name* **CHANGE** *field_name_old field_name_new* parametrs gde

- *table_name* — o'zgartirilayotgan ustun joylashgan jadval nomi;
- *field_name_old* — o'zgartirilayotgan ustun nomi;
- *field_name_new* — o'zgartirilayotgan ustun yangi nomi (agar ustun nomi o'zgartirilmasa *field_name_old* ga teng);
- parametrs — ustun yangi parametrlari.

Keyingi misolda field_1 tipini matn sifatida o'zgartiramiz:

$sql="ALTER TABLE my_table CHANGE field_1 field_1 TEXT";

mysql_query($sql)

Agar qo'shimcha ustun nomini field_2 deb o'zgartirish kerak bo'lsa:

$sql="ALTER TABLE my_table CHANGE field_1 field_2 TEXT";

mysql_query($sql)

Agar bir necha ustun xossalarini birdaniga o‘zgartirish lozim bo‘lsa, **CHANGE** *field_name_old field_name_new* parametrs konstruktsiyani vergul bilan har bir ustun uchun qaytaramiz:

$sql="ALTER TABLE my_table CHANGE field_1 field_2 TEXT,

CHANGE field_3 field_3 TEXT";

mysql_query($sql)

Jadval xossalarini o‘zgartirish: Ustunlarni o‘chirish (ALTER TABLE DROP).

Ustunni quyidagi konstruksiya yordamida o‘chirish mumkin:

ALTER TABLE *table_name* **DROP** *field_name*

Bu yerda:

- *table_name* — ustuni o‘chirilayotgan jadval nomi;
- *field_name* — o‘chirilayotgan ustun nomi.

$sql="ALTER TABLE search DROP id_num";

mysql_query($sql);

Agar biz birdaniga bir necha maydonlarni o‘chirmoqchi bo‘lsak, **DROP** *field_name* konstruktsiyani vergul bilan har bir ustun uchun qaytaramiz:

$sql="ALTER TABLE search DROP id_1, DROP id_2, DROP id_3";

mysql_query($sql)

Jadvalga satrlar qo‘shish (INSERT INTO).

Yozuvlarni joylash uchun **INSERT INTO** komandasidan foydalaniladi.

INSERT INTO *table_name*(field_name1, field_name2,...) values("content1", "content2",...)

Bu komanda *table_name* jadvaliga field_nameN maydonlariga contentN qiymat o‘rnatilgan yozuv qo‘shadi.

Masalan, agar biz manzilar va telefonlar (FISh, manzil, telefon) yaratmoqchi bo‘lsak, quyidagi kodni yozishimiz kerak:

CREATE TABLE tel_numb(fio text, address text, tel text)

tel_numb jadvaliga qiymatlarni quyidagicha joylash mumkin:

INSERT INTO tel_numb(fio, address, tel) Valiyevlar ("Valiyev Erkin", "Sebzor k., 18-uy", "260-23-23")

Joylash komandasida ko‘rsatilmagan maydonlar «aniqlanmagan» qiymatlar oladi (aniqlanmagan qiymat — bu bo‘sh satr emas, balki MySQL ga, shu maydonning net — hech qanday qiymati yo‘qligini bildiruvchi belgidir).

Agar jadval yaratilayotganda maydon NOT NULL bayrog‘i bilan belgilangan bo‘lsa va u yozuv joylashda qiymat olmasa, MySQL xato haqida ma’lumot qaytaradi.

Jadvalga binar ma’lumotlarni (apostrof va sleshlarni o‘z ichiga olgan satrlarni) joylashda ba’zi simvollar teskari sleshlar bilan, ya’ni '\' simvoli va nol kodga ega simvol bilan himoyalangan bo‘lishi kerak.

Jadvaldan satrlarni o‘chirish (DELETE FROM).

Yozuvni o‘chirish uchun **DELETE FROM** komandasi ishlatiladi.

DELETE FROM *table_name* WHERE (*ifoda*)

Bu komanda *table_name* jadvalidan *ifoda* bajarilgan hamma yozuvlarni o‘chiradi.

ifoda — bu oddiy mantiqiy ifoda.

Masalan FISh, manzil va telefonni o‘z ichiga olgan jadvaldan yozuvni o‘chirish:

DELETE FROM tel_numb WHERE (fish="Valiyev Erkin")

yoki, bir necha parametr bo‘yicha o‘chirish kerak bo‘lsa

DELETE FROM tel_numb WHERE (fio="Valiyev Erkin" && tel="260-23-23")

Ifodalarda maydonlarning nomlari, konstantalar va operatorlardan tashqari, sodda hisoblanuvchi qismlar kelishi mumkin, masalan: (id<10+4*5).

Bizda mehmonlar kitobi ma’lumotlarni saqlash uchun MySQL ma’lumotlar bazasidan foydalansin.

Jadval (nomi *db_guest*), qoldirilgan ma’lumotlarni o‘z ichiga olib, quyidagi tarkibga ega:

id — maydon yozuv identifikatsion unikal nomeri;

name — ma’lumot qoldirgan foydalanuvchi nomeri;

mail — foydalanuvchi e-mayli;

url — foydalanuvchi URL li;

content — ma’lumotning o‘zi.

Butun ma’lumotlar bazasini chiqaradigan va tanlangan ma’lumotni o‘chirishga imkon beradigan ssenariy (PHP da) yozamiz:

guest_delete.php fayli listingi

```
<html>
<head>
<title>Mehmonlar kitobi yozuvlarini o‘chirish.</title>
</head>
<body>
<?
// Ma’lumolar bazasiga ulanamiz
mysql_connect("localhost", "root", " ");
mysql_select_db(«test»);
// Agar o‘chirish tugmasini bosgan bo‘lsangiz
if(@$del_radio) {
// Tanlangan yozuvni o‘chiramiz
$sql="delete from db_guest where (id='$del_radio')";
mysql_query($sql);
};
// <I>$result</I> o‘zgaruvchiga butun qoldirilgan ma’lumotlar bazasini yozamiz
$sql="select * from db_guest";
$result=mysql_query($sql);
// Mehmonlar kitobida yozuvlar sonini aniqlaymiz
$rows=mysql_num_rows($result);
echo "<form method=get action='guest_delete.php'>";
echo "<table border=0 align=center>";
echo "<tr><td align=center><B>O‘chirish</B></td>";
echo "<td align=center><B>Ism</B></td>";
echo "<td align=center><B>E-mayl</B></td>";
echo "<td align=center><B>URL</B></td>";
echo "<td align=center><B>Ma’lumot</B></td></tr>";
for($i=0;$i<$rows;$i++) {
// Kursorni kerakli pozitsiyaga o‘rnatamiz:
mysql_data_seek($result,$i);
// assotsiativ massivga <I>$arr_guest</I>
// mehmonlar kitobi maydonlari qiymatini yozamiz:
$arr_guest=mysql_fetch_array($result);
echo "<tr><td>";
echo "<input type=radio name='del_radio' value=". $arr_guest["id"].""></td>";
echo "<td>".$arr_guest["name»]."</td>";
echo "<td>".$arr_guest["mail»]."</td>";
echo "<td>".$arr_guest["url»]."</td>";
echo "<td>".$arr_guest["content"]."</td></tr>";
};
```

```
echo "<tr><td colspan=5 align=center>";
echo "<input type=submit value='O‘chirish'>";
echo "</td></tr>";
echo "</table>";
echo "</form>";
?>
</body>
</html>
```

Jadvalda yozuvlarni yangilash (UPDATE).

Yozuvni yangilash uchun **UPDATE** komandasidan foydalaniladi

UPDATE *table_name* SET field_name1="var1", field_name2="var2",... WHERE (*ifoda*)

Bu komanda *table_name* jadvalidagi *ifoda* ga mos keluvchi hamma satrlar uchun, ko‘rsatilgan field_nameN maydoniga *varN* qiymat beradi.

Agar yozuvning hamma maydoni emas ba’zi maydonlarini yangilash lozim bo‘lgan holda bu komandadan foydalanish qulaydir.

Masalan, bizda mehmonlar kitobi ma’lumotni saqlash uchun MySQL ma’lumotlar bazasidan foydalansin.

Jadval (nomi *db_guest*), qoldirilgan ma’lumotlarni o‘z ichiga olib, quyidagi tarkibga ega:

id — yozuvning identifikatsion unikal nomeri;

name — ma’lumot qoldirgan foydalanuvchi nomi;

mail — foydalanuvchi e-mayli;

url — foydalanuvchi URLi;

content — ma’lumotning o‘zi.

Ma’umotlar bazasi hamma yozuvlarini chiqaruvchi va qoldirilgan ma’lumotlarga o‘zgartirish kiritishga imkon beruvchi ssenariy (PHP da) yozamiz:

guest_update.php fayli listingi

```
<html>
<head>
<title>Mehmonlar kitobidagi yozuvlarni yangilash.</title>
</head>
<body>
<?
// Ma’lumotlar bazasiga ulanamiz
mysql_connect("localhost", "root", "");
mysql_select_db("test");
```

```
// Agar yozuvni o‘zgartirish tugmasini bosgan bo‘lsak
if(@$submit_update) {
// Tanlangan yozuvni yangilaymiz
$sql=«update db_guest set name='$name', mail='$mail',
url='$url', content='$content'
where (id='$update')»;
mysql_query($sql);
}
// O‘zgaruvchiga <I>$result</I> butun qoldirilgan yozuvlar
bazasini yozamiz
$sql="select * from db_guest";
$result=mysql_query($sql)
// Mehmonlar kitobida yozuvlar sonini aniqlaymiz
$rows=mysql_num_rows($result);
echo "<table border=0 align=center>";
echo "<tr><td align=center><B>Ism</B></td>";
echo "<td align=center><B>E-mayl</B></td>";
echo "<td align=center><B>URL</B></td>";
echo "<td align=center><B>Ma’lumot</B></td>";
echo "<td align=center><B>O‘zgartirish</B></td>-
</tr>";
for($i=0;$i<$rows;$i++) {
// kursorni kerakli pozitsiyaga o‘rnatamiz:
mysql_data_seek($result,$i);
// Assotsiativ massivga <I>$arr_guest</I>
// mehmonlar kitobi maydonlari qiymatlarini yozamiz:
$arr_guest=mysql_fetch_array($result);
echo "<form method=get action='guest_update.php'>";
echo "<input type=hidden name='update' value=".
$arr_guest[«id»]."">";
echo "<tr>";
echo "<td><input type=text name='name' value=".
$arr_guest["name»]."></td>";
echo "<td><input type=text name='mail' value=".
$arr_guest["mail"]."></td>";
echo "<td><input type=text name='url' value=".
$arr_guest["url"]."></td>";
echo "<td><textarea name='content'>".$arr_guest["con-
tent"]."</textarea></td>";
echo "<td><input type=submit name='submit_update'
value='O‘zgartirilsin'></td>";
```

```
echo "</tr>";
echo "</form>";
};
echo "</table>";
echo "</form>";
?>
</body>
</html>
```

Jadvaldan yozuvlarni izlash (SELECT).

Jadvallardan izlash uchun **SELECT** komandasidan foydalaniladi:

SELECT * FROM *table_name* WHERE (*ifoda*) [order by *field_name* [desc][asc]]

Masalan, bizga mehmonlar kitobidagi ma'lumotlar saqlanuvchi jadvalda ma'lum foydalanuvchi qoldirgan hamma ma'lumotlarni topish kerak bo'lsin.

```
// Foydalanuvchi nomi
$user="Admin";
// name — mehmonlar kitobida ma'lumot qoldirgan foydalanuvchilar nomlari,
// saqlanuvchi jadvaldagi maydonning nomi,
// db_guest — mehmonlar kitobidagi ma'lumotlar saqlanuvchi jadval nomi,
$sql="select * from db_guest where (name='$user')";
$result=mysql_query($sql);
// Shartga mos keluvchi yozuvlar sonini aniqlaymiz
$rows=mysql_num_rows($result);
echo "$user mehmonlar kitobida $rows yozuvlarni qoldirdi."
```

Agar izlanayotgan matn butun maydonning hammasini emas, bir qismini egallasa nima qilish kerak (masalan, umumiy matn massivida so'z yoki jumla izlanganda)?

Masalan quyidagi komandadan foydalanish mumkin:

```
// $search — izlangan matnni o'z ichiga oladi
$sql="select * form db_guest where (locate('$search',content)>0)";
$result=mysql_query($sql)
```

Ma'nosi quyidagicha: agar $search satrlarining content maydoniga 0 dan katta bo'lsa(ya'ni ular umuman mavjud bo'lsa), yozuv topilgan yozuvlarga qo'shiladi.

Shuni aytish kerakki, ko'rsatilgan usul matnni registrni

hisobga olgan holda izlaydi. Registrni hisobga olmasdan izlash uchun quyidagi komandadan foydalanish lozim:

// $search — izlangan matnni o‘z ichiga oladi

$sql=″select * form db_guest where (locate(lower-('$search'),lower(content))>0)″;

$result=mysql_query($sql);

Ya’ni jadvaldagi izlanayotgan satr va yozuv yuqori registrga o‘tkaziladi, so‘ngra izlash amalga oshiriladi.

Bizda mehmonlar kitobining ma’lumotlar bazasi mavjud bo‘lsin va biz bu ma’lumotlarni qoldirilgan vaqti bo‘yicha tartiblashimiz kerak bo‘lsin.

Buning uchun baza maydonlaridan biri ma’lumot yozish vaqtini o‘z ichiga olishi kerak.

Biror ustun bo‘yicha tartiblash **order by** konstruktsiyasi yordamida amalga oshiriladi.

Bizning misolimizda «Yangiroq» ma’lumotlar yuqoriga joylashadi:

// time — yozish vaqtini o‘z ichiga olgan ustun

// format ″UNIX timestamp″

$sql=″SELECT * FROM db_guest ORDER BY data DESC″;

$result=mysql_query($sql)

Ma’lumotlar kamayish bo‘yicha tartiblangan.

Agar ma’lumotlarni o‘sish bo‘yicha tartiblash lozim bo‘lsa, **DESC** kalit so‘zi o‘rniga **ASC** qo‘llanadi:

$sql=″SELECT * FROM db_guest ORDER BY data ASC″;

$result=mysql_query($sql)

Ma’lumotlarni bir necha ustunlar bo‘yicha tartiblash mumkin, buning uchun ularning nomlari vergul bilan ajratib ko‘rsatilishi kerak.

Biror ustun bo‘yicha tartiblanayotganda, shu ustunda bir necha bir xil qiymatlar mavjud bo‘lsa kerak bo‘ladi.

$sql=″SELECT * FROM db_name ORDER BY field1,field2 ASC″;

$result=mysql_query($sql)

Ma’lumotlar avval birinchi ustun *field1* bo‘yicha tartiblanadi. So‘ngra agar birinchi ustunda bir necha bir xil qiymatlar bo‘lsa, ikkinchi ustun bo‘yicha qo‘shimcha tartiblash bajariladi (birinchi ustundagi bir qiymatli guruh ichida).

Agar izlashda hamma topilgan yozuvlar emas, ma’lum

guruhni chiqarish lozim bo‘lsa **LIMIT** parametridan foydalaniladi.

Bu parametrda ikki qiymat ko‘rsatiladi:

LIMIT *start,length*

Masalan, mehmonlar kitobi ma’lumotlar 20 dan 45 gacha yozuvlarni olish kerak bo‘lsin (ya’ni 25 ta yozuv chiqarish uchun):

```
$sql="select * from db_guest limit 20,25";
$result=mysql_query($sql);
```

Nazorat savollari

1. Parollar qanday ishlaydilar?
2. MySQL bilan ishlash qanday asosiy xususiyatlarga ega?
3. Serverdan versiyasi va vaqtni so‘rash uchun qanday komandani berish kerak?
4. Qanday tip 4 294 967 295 dan ortiq bo‘lmagan simvollarni saqlashi mumkin?
5. Binar ma’lumotlar — TEXT formatidagi ma’lumotlardan qanday farq qiladi?
6. MySQL ga qaysi baza bilan ishlash haqida ma’lumot berish uchun qanday komanda berish kerak?
7. Jadval yaratish CREATE TABLE komandasida qanday cheklanishlardan foydalanish mumkin?
8. Jadvalga yangi ustun qo‘shishni qanday konstruksiya yordamida amalga oshirish mumkin?
9. DESCRIBE Komandasi qanday vazifani bajaradi?

ADABIYOTLAR

1. *Роб П.* Системы баз данных: проектирование, реализация и управление (5-е издание). «БХВ — СПб».1200 стр., 2003 г.

2. *Попов И. И., Максимов Н. В., Голицына О. Л.* Базы данных. Издательство «Форум». 352 стр., 2004 г.

3. *Диго С.М.* Базы данных. Проектирование и использование. Издательство «Финансы и статистика». 592 стр., 2005 г.

4. *Дейт К.* Введение в системы баз данных. 8-е изд. «Вильямс». 1328 стр., 2005 г.

5. *Кузнецов С.Д.* Введение в стандарты языка баз данных SQL. M., 1998.

6. *Астахова И.Ф., Толстобров А.П.* SQL в примерах и задачах. Учебное пособие. «Новое знание», 176 стр., 2002 г.

7. *Полякова. Л.Н.* Основы SQL. Курс лекций. Учебное пособие. «ИНТУИТ.РУ». 368 стр., 2004 г.

8. *Бен Форта.* Освой самостоятельно SQL. 10 минут на урок (3-е издание). «Вильямс». 288 стр., 2005 г.

9. *Максим Кузнецов, Игорь Симдянов, Сергей Голышев.* PHP 5 на примерах. Серия: На примерах. «БХВ — СПб», 576 стр., 2005 г.

10. *Леон Аткинсон, Зеев Сураски.* PHP 5. Библиотека профессионала. Core PHP Programming. Серия: Библиотека профессионала. «Вильямс», 944 стр., 2005 г.

11. *Дмитрий Котеров, Алексей Костарев.* PHP 5. Серия: В подлиннике «БХВ — СПб», 1120 стр., 2005 г.

12. *Анатолий Мотев.* Уроки MySQL (+CD-ROM). Серия: Самоучитель. «БХВ — СПб», 208 стр., 2006 г.

13. *Люк Веллинг, Лора Томсон.* MySQL. Учебное пособие MySQL Tutorial. «Вильямс», 304 стр., 2005 г.

14. *Поль Дюбуа.* MySQL MySQL. Серия: Landmark. «Вильямс», 1056 стр., 2004 г.

SH. NAZIROV, A. NE'MATOV, R. QOBULOV

MA'LUMOTLAR BAZASINI DASTURLASH CHUQURLASHTIRILGAN KURSI

Axborot-kommunikatsiya texnologiyalari sohasidagi kasb-hunar kollejlarining «Axborot-kommunikatsiya tizimlari (3521916)» mutaxassisligi o'quvchilari uchun o'quv qo'llanma

«Sharq» nashriyot-matbaa
aksiyadorlik kompaniyasi
Bosh tahririyati
Toshkent — 2007

Muharrir *Z. Mirzahakimova*
Badiiy muharrir *F. Basharova*
Texnik muharrir *D. Gabdraxmanova*
Sahifalovchi *T. Ogay*

Bosishga ruxsat etildi: 15.10.2007. Bichimi: 60x90 $^{1}/_{16}$. «Tayms» garniturasi. Ofset bosma. Shartli bosma tobog'i 8,5. Nashriyot-hisob tobog'i 8,1. Adadi 1800 nusxa. 3992-son buyurtma. Bahosi kelishilgan narxda.

**«Sharq» nashriyot-matbaa
aksiyadorlik kompaniyasi bosmaxonasi,
100083, Toshkent shahri, «Buyuk Turon», 41.**