P. Luzanov, E. Rogov, I. Lyovshin
Tarjimon: F. Baltayev, Q. Xudaybergenov

Postgres 16

Birinchi tanishuv

Kirish

Ushbu kichik kitob – PostgreSQL bilan tanishishni boshlayotganlar uchun. Undan o'rganadiganlaringiz:

Kitobimiz PostgreSQL bilan bo'lgan birinchi tajribalaringizni yoqimli bo'lishiga va MBBT jamiyatiga kirishib ketishingizga yordam beradi deb umid qilamiz. Omad tilaymiz!

I PostgreSQL haqida

PostgreSQL – to'liq funksiyalari bilan ishlaydigan, erkin tarqatiladigan ochiq manbali ma'lumotlar bazasi. Akademik muhitda ishlab chiqilgan va uzoq tarixga ega bo'lib, uning atrofida ishlab chiquvchilarning keng hamjamiyatini birlashtirgan ushbu MBBT ko'pchilik mijozlar tomonidan talab qilinadigan barcha imkoniyatlarga ega. PostgreSQL butun dunyoda katta yuklama ostida ishlaydigan muhim biznes tizimlarini yaratish uchun faol qo'llaniladi.

Qisqacha tarix

Zamonaviy PostgreSQL kelib chiqish tarixi Berklidagi Kaliforniya universiteti professori Maykl Stounbreyker boshchiligida ishlab chiqilgan POSTGRES loyihasiga borib taqaladi. Bunga qadar Stounbreyker relyatsion ma'lumotlar bazasi tizimlari ichida birinchilardan bo'lgan INGRESni – ishlab chiqishga rahbarlik qilgan va POSTGRES ana shu loyihadagi tizimning qat'iy turlaridagi cheklovlarini yengib o'tish istagi tufayli yaralgan.

Loyiha ustida ish 1985 yilda boshlangan va 1988 yilgacha POSTQUEL so'rovlar tilining (o'sha paytda SQL hali umume'tirof etilgan standart bo'lmagan) ma'lumotlar modeli va ma'lumotlar bazasi dizaynini tavsiflovchi bir qator ilmiy maqolalar nashr etilgan.

POSTGRESni ba‘zida relyatsion bo‘lmagan MBBTlar qatoriga qo‘shishadi. Relyatsion modelning cheklovlari har doim tanqidga sabab bo‘lgan, garchi unda bir tomondan soddalik va qat‘iylik bo‘lsa ham. Biroq, kompyuter texnologiyalarining hayotning barcha sohalariga kirib borishi nostandart ma‘lumotlar turlarini va murakkab ob‘ektlarni meros qilib olish, yaratish va boshqarish kabi imkoniyatlarni qo‘llabquvvatlash uchun ilovalarning yangi sinflari hamda zarur ma‘lumotlar bazalarining paydo bo‘lishiga olib keldi.

Birinchi MBBT 1989 yilda ishga tushirilgan. Ma‘lumotlar bazasi bir necha yillar davomida takomillashtirildi va 1993 yilda 4.2 versiyasi chiqarilganda loyiha yopildi. Rasmiy halokatga qaramay, ochiq manba va BSD litsenziyasi Berkli bitiruvchilari Endryu Yu va Joli Chenga 1994 yilda uni yanada rivojlantirishga imkon berdi. Ular POSTQUEL so‘rovlar tilini o‘sha vaqtga kelib umumiy qabul qilingan SQL bilan almashtirdilar va loyihani Postgres95 deb nomladilar.

1996 yilga kelib, Postgres95 nomi vaqt sinovidan o‘tmasligi aniq bo‘ldi va yangi – PostgreSQL nomi tanlandi. Bu nom o‘zida ham asl POSTGRES loyihasining atamasini, ham SQLga o‘tishni aks ettirdi. Tan olish kerakki, nomni talaffuz qilish qiyin bo‘lib chiqdi, uni o‘qishda: PostgreSQL “postgres-kuel” yoki oddiygina “postgres” deb talaffuz qilinishi kerak, lekin “postgre” emas.

Yangi talqin 6.0 sifatida boshlandi va asl raqamlashni davom ettirdi. Loyiha o‘sib bordi va uni boshqarish dastlab PostgreSQLni ishlab chiquvchilarning global guruhi deya nom olgan (PostgreSQL Global Development Group) proaktiv foydalanuvchilar va ishlab chiquvchilarning kichik guruhi tomonidan o‘z zimmasiga olindi.

Taraqqiyot

Rivojlanish rejalari va yangi talqinlarni chiqarish bo'yicha barcha asosiy qarorlar hozirda yetti kishidan iborat boshqaruvchi qo'mita (Core team) tomonidan qabul qilinadi.

Tizim rivojiga o'z hissasini qo'shayotgan oddiy dasturchilardan tashqari, PostgreSQL rivojiga katta hissa qo'shayotgan asosiy (major contributors) va manba kodlari bazasiga yozish huquqiga ega bo'lgan (committers) ishlab chiquvchilar guruhi ham mavjud. Guruhlarning tarkibi vaqt o'tishi bilan o'zgaradi, yangi a'zolar paydo bo'ladi, ba'zilari loyihani tark etadilar. Ishlab chiquvchilarning joriy ro'yxati rasmiy veb-saytda e'lon qilingan: postgresql.org/community/contributors.

Rossiyalik ishlab chiquvchilarning PostgreSQLga qo'shayotgan hissasi ahamiyatga sazovordir. Bu Rossiya ishtirok etgan eng katta ochiq kodli loyihalardan biri.

PostgreSQL rivojida boshqaruvchi qo'mita a'zosi bo'lgan Krasnoyarsklik dasturchi Vadim Mixeev mehnatlari muhim ahamiyatga ega. U bir vaqtda ma'lumotlardan foydalanishning ko'ptalqinli boshqaruvni ta'minlash (MVCC), tozalash tizimi (vacuum), tranzaksiyalar jurnali (WAL), qism so'rovlar, triggerlar kabi muhim hisoblangan qismlar muallifi. Hozirda u loyiha bilan shug'ullanmaydi.

2015 yilda Oleg Bortunov, astronom va "ГАИШ МГУ" ilmiy xodimi Fedor Sigayev va Aleksandr Korotkovlar bilan PostgreSQL Professional kompaniyasiga asos soldi.

Ular tomonidan bajarilgan ishlar yo'nalishi sifatida PostgreSQLni mahalliylashtirish (mahalliy kodlarni va Unicodeni qo'llab – quvvatlash), umummatn qidiruv tizimi, massivlar va zaiftuzilmali ma'lumotlar bilan ishlash (hstore, json, jsonb),

indekslash yangi usullari (GiST, SP-GiST, GIN va RUM, Bloom) larni aytish mumkin. Shuningdek ular ko'p sondagi mashhur kengaytmalar mualliflaridir.

PostgreSQLning keyingi talqini uchun ishlash davri odatda bir yil davom etadi. Bu vaqt ichida tuzatishlar, o'zgartirishlar va yangi funksiyalarga ega patchlar ko'rib chiqish uchun hammadan qabul qilinadi. An'anaga ko'ra, pgsql-hackers pochta ro'yxati patchlarni muhokama qilish uchun ishlatiladi. Agar hamjamiyat g'oyani foydali deb bilsa, uni amalga oshirishni to'g'ri deb topsa va kod boshqa ishlab chiquvchilar majburiy sinovidan o'tgan bo'lsa, unda patch relizga kiritiladi.

Ma'lum bir vaqtda (odatda bahorda, relizdan taxminan olti oy oldin) kodni barqarorlashtirish bosqichi e'lon qilinadi – yangi funksionallik keyingi talqingacha qoldiriladi va faqat relizga kiritilgan patchlarga tuzatishlar yoki yaxshilanishlar qabul qilinadi. Reliz sikli davomida bir necha marta beta-talqinlari chiqariladi, sikl oxirida reliz nomzodi (major) paydo bo'ladi va tez orada PostgreSQLning yangi asosiy talqini chiqariladi.

Ilgari asosiy talqin raqami ikkita raqamdan iborat edi, ammo 2017 yildan boshlab faqat bitta raqamni qoldirishga qaror qilindi. Shunday qilib, 9.6dan keyin 10 keldi va PostgreSQLning oxirgi joriy talqini 2023 yil sentyabr oyi o'rtalarida chiqarilgan 16talqinidir.

MBBTning yangi talqini ustida ishlayotganda xatolar aniqlanishi mumkin. Ularning eng tanqidiylari nafaqat joriy, balki oldingi talqinlarda ham tuzatiladi. Odatda, to'plangan tuzatishlarni qamragan qo'shimcha (minor) talqinlar har chorakda bir marta chiqariladi. Masalan, 12.5 talqinida faqat 12.4da topilgan xatolarning tuzatishlari mavjud va 16.1da faqat 16.0da topilgan xatolar uchun tuzatishlar mavjud.

Qo'llab-quvvatlash

Global PostgreSQL ishlab chiqish jamoasi tizimning asosiy talqinlarini chiqarilgan kundan boshlab besh yil davomida qo'llab-quvvatlaydi. Ushbu qo'llab-quvvatlash, shuningdek, rivojlanishni muvofiqlashtirish pochta ro'yxatlari orqali taqdim etiladi. To'g'ri formatlangan xato xabari imkon qadar tezroq hal qilish uchun barcha imkoniyatlarga ega: ko'pincha xatolarni tuzatish 24 soat ichida chiqariladi.

Ishlab chiquvchilar hamjamiyatining yordamiga qo'shimcha ravishda, butun dunyo bo'ylab bir qator kompaniyalar PostgreSQL uchun tijoriy yordam ko'rsatadilar. Rossiyada bunday tashkilotlardan Postgres Professional (postgrespro.ru) bo'lib, ular 24x7 qo'llab-quvvatlash xizmatlarini taqdim etadi.

Zamonaviy holati

PostgreSQL dunyodagi eng nom taratgan ma'lumotlar bazasi tizimlaridan biridir. Yigirma yildan ortiq vaqt davomida mustahkam ilmiy asoslarga tayanib, to'laqonli MBBTga aylandi. Shu bilan birga u tashkilot miqyosida foydalanish uchun mos bo'lgan hamda tijorat tizimlariga haqiqiy alternativ hisoblanadi. Bunga ishonch hosil qilish uchun PostgreSQL 16ning so'nggi talqinining eng muhim xususiyatlarini ko'rib chiqing.

Ishonchlilik va barqarorlik

Korxona ilovalarida muhim ma'lumotlar bilan ishlashda ishonchlilik ayniqsa muhimdir. Shu maqsadda PostgreSQL

sizga “issiq” zaxira nusxalarini sozlash, o‘tmishdagi ma‘lum bir vaqt holatini tiklash va turli xil replikatsiya turlarini (sinxron, asinxron, kaskad) sozlash imkonini beradi.

Xavfsizlik

PostgreSQL foydalanuvchilarga xavfsiz SSL orqali ulanish imkonini beradi. Parolni autentifikatsiya qilish (shu jumladan SCRAM), mijoz sertifikatlaridan foydalanish va tashqi xizmatlar (LDAP, RADIUS, PAM, Kerberos) yordamida autentifikatsiya qilish ham mumkin.

Ma‘lumotlar bazasi obyektlaridan foydalanish imkonini boshqarish uchun quyidagi variantlar taqdim etiladi:

- foydalanuvchi hisoblari va guruh rollarini yaratish va boshqarish;
- individual foydalanuvchilar va guruhlar darajasida ma‘lumotlar bazasi obyektlaridan foydalanish imkonlarini farqlash;
- alohida ustunlar va qatorlar darajasida foydalanish imkonini batafsil nazorat qilish;
- SE-PostgreSQL funksiyasi (kirishni “majburiy” boshqarish) orqali SELinuxni qo‘llab-quvvatlash.

Postgres Professional kompaniyasi tomonidan yaratilgan PostgreSQL maxsus talqini – Postgres Pro Certified, shaxsiy ma‘lumotlar va maxfiy axborotlar bilan ishlovchi tizimlar uchun himoyalangan “ФСТЭК” sertifikatiga ega.

Standartlarga moslik

PostgreSQL yangi ANSI SQL standart talablari ishlab chiqilsa ularga bo'lgan moslikni ta'minlaydi. Bu SQL-92dan to JSON formati bilan ishlashni qo'llab-quvvatlashni standartlashtirgan so'nggi SQL:2016 gacha bo'lgan standartning barcha talqinlari uchun ham amal qiladi. Ushbu funksionalning muhim qismi allaqachon PostgreSQL 16da joriy qilingan.

Umuman olganda, PostgreSQL standartga yuqori darajadagi muvofiqlikni ta'minlaydi va 177 talab qilinadigan imkoniyatlardan 170 tasini, shuningdek, ko'p sonli qo'shimcha imkoniyatlarni qo'llab-quvvatlaydi.

Tranzaksiyani qo'llab-quvvatlash

PostgreSQL ACID xususiyatlarini va tranzaksiyalarni samarali izolyatsiya qilishni to'liq qo'llab-quvvatlaydi. Bunga Multi-Version Concurrency Control (MVCC) yordamida erishiladi. Bu barcha hollarda qator bloklanishlaridan qochadi, bundan tashqari bir nechta jarayonlar bir vaqtning o'zida bir xil ma'lumotlar qatorini o'zgartiradi: o'qish hech qachon yozishni bloklamaydi va yozish esa – o'qishni.

Xuddi shu narsa Serializable Snapshot innovatsion izolyatsiya tizimidan foydalangan holda ishlovchi eng qat'iy izolyatsiyalash darajasiga tegishli bo'lib, seriyalashdan kelib chiqish mumkin bo'lgan xatoliklardan batamom halos qiladi hamda ketma-ket va parallel bajarilishlar natijalarining mosligini ta'minlanishini kafolatlaydi.

Ilova yaratuvchilar uchun

Ilovalarni ishlab chiquvchilar ixtiyorida har qanday turdagi ilovalarni amalga oshirish imkonini beruvchi boy uskunalar to'plami mavjud:

- turli server dasturlash tillari: ichida o'rnatilgan PL/pgSQL (SQL bilan yaqindan integratsiyalashuvi tufayli ancha qulay), tezkorlikka moyil muhim vazifalar uchun C, Perl, Python, Tcl, shuningdek JavaScript, Java va boshqalar;
- istalgan tildagi ilovalardan ma'lumotlar bazasiga murojaat uchun dasturiy interfeyslar, shu jumladan standart ODBC va JDBC interfeyslari;
- server tomonida har qanday murakkablikdagi mantiqni samarali amalga oshirish imkonini beruvchi ma'lumotlar bazasi ob'ektlari to'plami: jadvallar va indekslar, ketmaketliklar, yaxlitlik cheklovlari, ko'rinishlar va moddiylashtirilgan ko'rinishlar, bo'limlar, qism so'rovlar (shu jumladan rekursiv), yig'ish va oyna funktsiyalari, saqlanuvchi funktsiyalar, triggerlar va boshqalar;
- rus va barcha Yevropa tillarini qo'llab-quvvatlaydigan, indeksli foydalanish bilan samarali to'ldirilgan moslashuvchan umummatnli qidiruv tizimi;
- NoSQL-ga xos zaif tuzilmali ma'lumotlar: hstore (kalit-qiymat juftliklari bazasi), xml, json (ham matnda, ham yanada samarali ikkilik tasviri ko'rinishi jsonb-da);
- ma'lumotlar manbalarini, barcha asosiy ma'lumotlar bazalarini ham o'z ichiga olgan holda, SQL/MED standarti bo'yicha tashqi jadvallar sifatida ulardan to'liq foydalanish imkoniyati bilan ularga yozish va tarqatilgan holda so'rovlarni bajarish uchun ulash (Foreign Data Wrappers).

Masshtablilik va ishlash tezligi

PostgreSQL zamonaviy ko'p yadroli protsessor arxitekturasidan samarali foydalanadi – MBBT ishlashi yadrolar sonining ko'payishi bilan deyarli chiziqli bog'liq holda o'sadi.

So'rovlarni va ba'zi xizmat buyruqlarini bajarish uchun parallel rejim mavjud (masalan, indekslarni yaratish va tozalash). Ushbu rejimda ma'lumotlarni o'qish va ulanish operatsiyalari bir vaqtning o'zida ishlaydigan bir nechta jarayonlar tomonidan amalga oshiriladi. So'rovlarni JIT-kompilyatsiya qilish – apparat bilan bog'liq jarayonlarni tezlashtirish qobiliyatini yaxshilaydi. PostgreSQLning har bir talqinida yangi parallellashtirish xususiyatlari paydo bo'lib kelayabdi.

Gorizontal masshtablash uchun PostgreSQL ham fizik, ham mantiqiy replikatsiyani ta'minlaydi. Bu – nosozliklarga chidamlilik, yuqori unumdorlik va geografik taqsimotni ta'minlash uchun, PostgreSQL asosida klasterlar yaratish imkonini beradi. Bunday tizimlarga misollar: Citus (Citusdata), Postgres-BDR (2ndQuadrant), Multimaster va BiHA (Postgres Professional), Patroni (Zalando).

So'rovlarni rejalashtiruvchi

PostgreSQL xarajatlarga asoslangan so'rovlarni rejalashtiruvchidan foydalanadi. Statistik ma'lumotlarni to'plash shuningdek disk va protsessor operatsiyalari uchun resurslar sarfini hisobga olish va hatto eng murakkab so'rovlarni optimallashtirishga imkon beradi. Rejalashtiruvchi ixtiyorida ilg'or tijorat MBBTlarida mavjud bo'lgan ma'lumotlarni olib chiqish va ulardan foydalanishning barcha usullari.

Indekslash imkoniyatlari

PostgreSQL turli xil indekslash usullarini qo‘llaydi. An‘anaviy B-daraxtlardan tashqari, bir qator boshqa foydalanish usullari mavjud.

- Hash – indeks, xeshlashga asoslangan. B-daraxtdan farqli o‘laroq faqat tenglikni tekshirishda ishlaydi, lekin ba‘zi hollarda u ixchamroq va samarali bo‘lib chiqadi.
- GiST – bu tartiblash imkonini bermaydigan ma‘lumotlar uchun ishlatiladigan umumlashtirilgan muvozanatli qidiruv daraxti. Misollar sifatida, eng yaqin qo‘shnilarni tez qidirish (k-NN qidiruvi) imkoniyati bilan tekislikdagi nuqtalarni indekslash uchun xizmat qiladigan R-daraxtlarni va intervallarning kesishishini indekslash amallarini keltirish mumkin.
- SP-GiST – umumlashtirilgan muvozanatlshtirilmagan daraxt, qiymatlar sohasini kesishmaydigan qism sohalarga ajratishga asoslangan. Misol tariqasida fazoviy ma‘lumotlar uchun kvadrantlar daraxti va matnli qatorlar uchun old qo‘shimchali daraxt bo‘lishi mumkin.
- GIN – umumlashtirilgan invertlangan indeks, elementlardan tashkil topgan murakkab qiymatlar uchun ishlatiladi. Uning asosiy ishlatilish maydoni – umummatn qidiruvi, ya‘ni so‘rovda ko‘rsatilgan so‘z uchraydigan barcha hujjatlarni topish. Qo‘llanilishiga yana boshqa bir misol sifatida massiv ichidan qiymatni qidirishni olish mumkin.
- RUM – umummatn qidiruvi uchun GIN uslubining keyingi rivojlantirilgan talqini. Ushbu indeks kengaytma sifatida bo‘lib iboraviy qidirishlarni tezlashtiradi va natijani muvofiqligi bo‘yicha hech qanday qo‘shimcha hisoblashlarsiz tartiblaydi.

- BRIN – indeks o'lchami va qidiruv tezligi o'rtasida kelishuvga imkon beruvchi ixcham tuzilma. Ushbu indeks katta klasterli jadvallarda samarali.
- Bloom – Bloom filtriga asoslangan indeks. Juda ixchamligi tufayli, bu sizga aniq keraksiz qatorlarni tezda olib tashlashga imkon beradi, ammo qolganlarini qayta tekshirishni talab qiladi.

Ko'p turdagi indekslarni faqat bitta emas, balki jadvalning bir nechta ustunlarida ham yaratish mumkin. Turi qanday bo'lishidan qat'iy nazar, siz indekslarni ham ustun bo'yicha, ham ixtiyoriy ifodalarda qurishingiz mumkin, shuningdek, faqat ma'lum qatorlar uchun qism indekslarni yaratishingiz mumkin. Qatorlarni qamrab olgan indekslar jadvalga kirmasdan indeksning o'zidan barcha kerakli ma'lumotlarni olish orqali so'rovlar bajarilishini tezlashtirish imkonini beradi.

Rejalashtiruvchining qurollari qatoriga kiruvchi bit xarita bo'yicha skanerlash bir nechta indekslarni birlashtirib ma'lumotlarni olib chiqish tezligini oshiradi.

Krossplatformalilik

PostgreSQL Unixlar oilasiga mansub operatsion tizimlarda, jumladan Linux, FreeBSD, Solaris va macOS, shuningdek, Windowsning server va mijoz talqinlarida ishlaydi.

C tilida ochiq va ko'chma bo'lganligi sababli, PostgreSQLni turli xil platformalarda hatto PostgreSQL jamiyati tomonidan yig'ma paketi qo'llab-quvvatlanishi bo'lmagan platformalarga ham, yig'ish mumkin.

Kengayuvchanlik

Kengayuvchanlik – PostgreSQL arxitekturasi asosidagi tizimning asosiy afzalliklaridan biridir. Foydalanuvchilar asosiy tizim kodini o'zgartirmasdan, mustaqil ravishda quyidagilarni qo'shishlari mumkin:

- ma'lumotlar turlari;
- yangi turlar bilan ishlash funksiyalari va operatorlari;
- indeks va jadvalga kirish usullari;
- server dasturlash tillari;
- tashqi ma'lumotlar manbalariga ulanishlar (Foreign Data Wrappers);
- yuklab olinadigan kengaytmalar.

Kengaytmalarni to'liq qo'llab-quvvatlash PostgreSQL yadrosiga o'zgartirishlar kiritmasdan va kerak bo'lganda ulanishga ruxsat bermasdan har qanday murakkablikdagi funksionallikni amalga oshirish imkonini beradi. Masalan, kengaytmalar shaklida yaratilgan bir qancha murakkab tizimlar mavjud:

- CitusDB – turli PostgreSQLga ma'lumotlarni tarqatish (parchalash-sharding) va so'rovlarni parallel ravishda bajarish qobiliyati.
- PostGIS – geografik axborot ma'lumotlarini qayta ishlash uchun eng mashhur va kuchli tizimlardan biri;
- TimescaleDB – vaqt qatorlari, shu jumladan maxsus qismlarga ajratish (partition) va parchalash (sharding) bilan ishlash.

PostgreSQL 16 yig'masi o'z ichiga olgan standart to'plamning o'zi, ishonch va foydaliligini isbotlagan ellikga yaqin kengaytmalarga ega.

Mumkinlik

BSD va MIT litsenziyalariga o‘xshash liberal PostgreSQL litsenziyasi MBBTdan cheklovsiz foydalanish, kodni o‘zgartirish va boshqa mahsulotlarga, shu jumladan xususiy va tijoriy mahsulotlarga qo‘shish imkonini beradi.

Bog‘liqsizlik

PostgreSQL biror tashkilotga qarashli emas va u xalqaro jamiyat tomonidan rivojlanadi. Bu shuni anglatadiki, PostgreSQLdan foydalanadigan tizimlar har qanday sotuvchidan mustaqil bo‘lib, ularga kiritgan investitsiyalaringiz har qanday vaziyatda ham himoyalangandir.

II PostgreSQL 16 yangiliklari

Agar siz PostgreSQLning oldingi talqinlari bilan tanish bo‘lsangiz, ushbu bob sizga o‘tgan yil ichida nima o‘zgarganligi haqida ma‘lumotlar beradi. Faqat ba‘zi o‘zgarishlargina bu yerda keltirilgan. To‘liq ro‘yxat uchun nashr eslatmalariga qarang: postgrespro.ru/docs/postgresql/16/release-16. Postgres Pro tomonidan yuritiladigan Habrdagi blogda yangi mahsulotlar bilan tezda tanishishingiz mumkin: habr.com/ru/companies/postgrespro/articles/.

SQL buyruqlari

`FROM` bandidagi qism so‘rovlar uchun taxalluslar endi ixtiyoriy bo‘lib, ular foydali bo‘lmagan taqdirda nom berish zaruriyatini olib tashlash imkonini yaratdi.

Endilikda **kengaytirilgan statistika uchun nom o‘ylab topishga** hojat yo‘q. `CREATE INDEX` buyrug‘i nomlashni boshqara oladi va `CREATE STATISTICS` ham huddi shunday.

O‘zgarishlar **`REINDEX` buyrug‘iga** ham ta‘sir qildi: ma‘lumotlar bazasi nomini belgilash ixtiyoriy bo‘lib qoldi va `REINDEX DATABASE` opsiyasi endi tizim katalogi indekslariga ta‘sir o‘tkazmaydi.

`CREATE TABLE` buyrug'i endi **TOAST saqlash strategiyasini belgilash** imkoniyatiga ega. Ilgari, `ALTER` yordamida jadval ifodalanishini o'zgartirishingiz kerak edi.

Yangi SQL:2023 standartida ko'rsatilganidek, **butun son konstantalari nafaqat o'nlik, balki o'n oltilik, sakkizlik yoki ikkilik tizimlarda ham yozilishi mumkin**, bunda raqamlar guruhlarini _ vizual ajratish pastki chiziq yordamida amalga oshiriladi.

`COPY FROM` buyrug'i ma'lum kirish qiymatlarini **ustunning sukut (default) qiymati bilan almashtirishni** o'rgandi.

Funksiyalar

SQL/JSON standartini joriy qilish bo'yicha qiyin ishlar davom etmoqda. Ushbu nashr `json_array`, `json_arrayagg`, `json_object`, `json_objectagg` **konstruktorlar funktsiyalari** va `IS JSON`, `IS JSON VALUE`, `IS JSON ARRAY`, `IS JSON OBJECT`, `IS JSON SCALAR` `predikatlarini` qo'shdi. Eng qiziqarli narsa amalga oshirilmagan bo'lib qolmoqda _ `json_table` funksiyasi, biz uni ham kutib qolamiz.

Taqsimlangan tasodifiy sonlarni olish uchun yangi `random_normal` funksiyasi, shu bilan birga uning uchun `erf` xato funksiyasi va qo'shimcha xato funksiyasi `erfc` yaratildi.

`array_shuffle` funksiyasi asl massivning elementlarini aralashtirib yuboradi va `array_sample` asl massivning elementlaridan ko'rsatilgan soncha olinadigan **tasodifiy elementlar** soniga ega massivni qaytaradi.

SQL standartida tasvirlangan `any_value`, yangi agregat funksiyasi – **qatorlar guruhi uchun ixtiyoriy bo'sh bo'lmagan qiymatni qaytaradi**.

Yangi `date_add` va `date_subtract` funktsiyalari sanalarni qo'shish va ayirishda **vaqt mintaqasini aniq ko'rsatish** imkonini beradi.

`pg_split_walfile_name` funksiyasi **WAL-fayl nomini tashkillashtiradi va offsetni LSN pozitsiyasiga suradi**.

`pg_input_is_valid` funktsiyasi birinchi argument sifatidagi **kirish qiymatining ikkinchi argument – ma'lumotlar turi uchun mumkin bo'lgan qiymat yoki yo'qligini tekshiradi**. Xato haqidagi batafsil ma'lumotlar `pg_input_error_info` funktsiyasi tomonidan taqdim etiladi.

Mantiqiy replikatsiya

Mantiqiy replikatsiya faol rivojlanishda davom etmoqda. **Mantiqiy replikatsiya jismoniy replikatsiya** bilan birga ishlay olishni boshladi, bu esa qo'llab-quvvatlanadigan foydalanish holatlari doirasini yanada kengaytiradi.

Ikki tomonlamali mantiqiy replikatsiya, albatta, hali multimaster degani emas (pgconf.ru/2023/341014), lekin bu ushbu yo'nalishdagi muhim qadamdir.

Agar jadvalda unikal indeks bo'lmasa, replikatsiya **unikal bo'lmagan indeksdan** foydalanishi mumkin.

Ikkilik formatdagi dastlabki sinxronizatsiya **ma'lumotlar uzatishni tezlashtiradi** va uzoq davom etadigan tranzaktsiyalardagi o'zgartirishlar nashriyot serverida fiksirlashlarni kutmasdan **parallel ravishda qo'llanilishi** mumkin.

Obuna yaratish huquqlari superfoydalanuvchidan yangi `pg_create_subscription` maxsus roliga o'tkazildi. Va sukut

bo‘yicha, **o‘zgarishlarni qo‘llash** avvalgidek obuna egasi emas, balki jadval egasi nomidan qo‘llaniladi.

Tozalash

Muzlatish to‘g‘risidagi qaror avvalgidek satr bo‘yicha emas, **balki butun sahifa uchun qabul qilinadi**: agar siz satrning bitta talqinini muzlatishingiz kerak bo‘lsa, qolganlarini muzlatishingizga hech narsa to‘sqinlik qilmaydi. Shu bilan birga, to‘liq tasviri jurnalga qayd etilayotgan sahifalar ham muzlatiladi.

vacuum_defer_cleanup_age **parametri olib tashlandi**. Bu muammolarni keltirib chiqardi, ammo qulaylik yaratmadi: katta qiymatlar jadvallarning o‘sishiga olib keldi, kichiklari esa replikada WALdan foydalanishdagi nizolarni oldini olishga yordam bermadi.

Endi yangi *vacuum_buffer_usage_limit* parametrida yoki VACUUM va ANALYZEni chaqirganda **tozalash buferining o‘lchami** ko‘rsatilishi mumkin. Ilgari o‘lcham fiksirlangan edi.

Qo‘lda tozalash sizga kerakli sxemalardan jadvallarni tanlash, toast-jadvallarini alohida qayta ishlash va ma‘lumotlar bazasi statistikasini yangilash bosqichini boshqarish imkonini beradi.

Monitoring

Eng muhim yangilik — **`pg_stat_io` ko‘rinishi bo‘lib, unda buferlar soni va ishlash vaqtini o‘z ichiga olgan batafsil kiritish-chiqarish statistikasi mavjud**. Endi, masalan, bufer keshidan

diskka ma'lumotlarni kim yozishini aniq tushunish mumkin bo'ladi. Ilgari bu mumkin emas edi, chunki `pg_stat_bgwriter` ko'rinishida mijoz jarayonlari va tozalash o'rtasidagi farqni bila olmasdik.

Mavjud statistik ko'rinishlarda ham o'zgarishlar mavjud. `n_tup_newpage_upd` ustuni `pg_stat_*_tables` oilasiga qo'shildi, bu sizga **HOT optimallashtirish samaradorligini baholash imkonini** beradi. **DDL buyruqlarini normallashtirish** paydo bo'ldi, endi `pg_stat_statements`da dublikatlar bo'lmaydi. `pg_stat_all_tables` va `pg_stat_all_indexes` nafaqat skanerlashlar sonini, balki indekslar va jadvallarga **oxirgi marta kirishni ham kuzatib boradi**.

`EXPLAIN` buyrug'i `$1`, `$2` va xokozolarga almashtirilgan parametrlar bilan **umumiy so'rov rejasini** ko'rsatishi mumkin.

`auto_explain` moduli **parametr qiymatlarini** (*log_parameter_max_length*) va **so'rov identifikatorlarini** (*log_verbose*) jurnalga yozishni o'rgandi.

Xulosa ma'lumotlarini tezda olish uchun `pg_buffercache` kengaytmasiga ikkita yangi `pg_buffercache_usage_counts` va `pg_buffercache_summary` funksiyalari qo'shildi.

Optimallashtirishlar

BRIN indekslari endi HOT yangilanishlariga to'sqinlik qilmaydi, chunki ular ma'lum qator talqinlariga emas, balki sahifa diapazonlariga murojaat qiladi. Oldingi, 15-talqindagi urinish yuzaga kelgan qiyinchiliklar tufayli bekor qilingandi, ular endi bartaraf etildi.

Rejalashtiruvchi **o'ng anti-ulanishni** ko'rib chiqa boshladi, ilgari unga faqat chap tomonnikidan foydalanish imkoni mavjud edi.

To'liq va o'ng xesh-birikmalari parallel ravishda amalga oshirilishi mumkin.

Oyna chegarasi turini optimal tanlash tufayli standart oyna funksiyalarini (masalan, `row_number` yoki `rank`) bajarilishi tezlashdi.

Agregat funksiyalar chaqirilishida `ORDER BY` yoki `DISTINCT` saralash bandi ko'rsatilganda tartiblangan ma'lumotlarni olish uchun indeksdan foydalanishi mumkin bo'ldi. Shuningdek `string_agg` va `array_agg` agregat funktsiyalari endi **parallel ravishda** bajariladi.

Endilikda `GROUPBY` va `DISTINCT` bandlaridagi takroriy ustunlar unumdorlikni pasaytirmaydi: bu optimallashtirish avtomatik generatsiya qilingan so'rovlar uchun foydali bo'lishi mumkin.

Xesh-indekslarini yaratish ma'lumotlarni faqat savat raqami bo'yicha emas, balki qiymat bo'yicha ham oldindan saralash orqali tezlashdi.

Qatorlar qo'shiladigan bo'lim nomini keshlash orqali qismlarga bo'lingan (partition) jadvalga ommaviy kiritish tezlashtirildi.

Sur'atlardan qidirish – SIMD ko'rsatmalari yordamida optimallashtirilgan bo'lib, u bir vaqtning o'zida yuzlab yozish tranzaktsiyalari bajarilganda ishlashga foyda keltiradi.

Munosabatlar kengaytmasi sahifalar qo'shilganda blokirovkani kam vaqtga ushlab turadi va shu orqali jadvalga bir vaqtda faol ma'lumotlar qo'shilishi mavjud bo'lgandagi ishlashni tezlashtiradi.

force_parallel_mode **opsiyasi nihoyat** ***debug_parallel_query*** **deb o‘zgartirildi** va u tajribasiz administratorlarni mavjud bo‘lmagan vasvasaga tushishdan to‘xtatadi degan umiddamiz.

Foydalanish nazorati

Ishlab chiquvchilar rollar va imtiyozlar tizimini faol ravishda qayta ko‘rib chiqmoqdalar.

`GRANT` buyrug‘ining sintaksisi kengaytirildi. Endi, bir rolni boshqasiga qo‘shganingizda, **imtiyozlar meros qilib olinishi kerakmi** (INHERIT OPTION) va **rolga o‘tishga ruxsat beriladimi** (SET OPTION) kabi xossalarni belgilashingiz mumkin. Rol a‘zoligi uchinchi rol nomidan berilsa, server harakati endi shunga mos bo‘ladi.

`CREATEROLE` atributining imkoniyatlari faqat o‘zingizga tegishli bo‘lgan narsalarni uzatishingiz mumkin degan tamoyilga muvofiq qisqartirildi.

Klasterni ishga tushirish vaqtida yaratilgan foydalanuvchidan **superuserlik bekor qilinmaydi**.

Yangi `SYSTEM_USER` funktsiyasi SQL standarti bilan moslandi, shu bilan birga u **autentifikatsiya usuli** va **tashqi foydalanuvchi** haqidagi ma‘lumotlarni ham ko‘rsatadi.

Konfiguratsiya fayllarini qayta ishlashda ma‘lum bir birlashmalar bo‘ldi. pg_hba.conf va pg_ident.conf fayllari, postgresql.conf bilan bir qatorda, endi boshqa fayllarni kiritish uchun **include direktivalarini** qo‘llab-quvvatlashni boshlashdi va `pg_hba_file_rules` va `pg_ident_file_mappings` ko‘rinishlarida fayl nomi va qoida yoki moslik raqamiga ega

ustunlar paydo bo‘ldi. pg_ident.conf faylida PostgreSQL foydalanuvchi nomi endi pg_hba.conf bilan bir xil qoidalarga muvofiq qayta ishlanadi: siz **all** va **guruh rol nomlarini** belgilashingiz va **regulyar iboralardan** foydalanishingiz mumkin. Regulyar ifodalar pg_hba.confda foydalanuvchi nomlari va ma‘lumotlar bazalari uchun ham qo‘llab-quvvatlanadi.

Yangi *reserved_connections* parametridan foydalanib, siz yangi *pg_use_reserved_connections* sobit roliga kiritilgan rollar tomonidan ishlatilishi mumkin bo‘lgan ulanishlarning kerakli sonini **zaxiraga olishingiz mumkin**. *superuser_reserved_connections* zaxirasi ham saqlanib qoladi, lekin eng ekstremal holatlar uchun.

Endi libpq kutubxonasi ulanish balansini qo‘llab-quvvatlaydi (`load_balance_hosts`) va qabul qilinadigan **autentifikatsiya usullarini** (`require_auth`) belgilash qobiliyatiga ega.

Yangi *scram_iterations* parametrini oshirish autentifikatsiyani sekinlashtirsada, **saqlangan parollarning turg‘unligini oshiradi**.

Utilitlar va kengaytmalar

psql

Endi **sarlavhaning kengligini kengaytirilgan formatda (`\pset xheader_width`) boshqarish mumkin**.

Bo‘lingan jadval uchun `\d+` buyrug‘i bo‘limlarni, jumladan uchinchi tomon jadvallarini — `FOREIGN` so‘zi bilan belgilab ko‘rsatadi.

Yangi `\bind` buyrug‘i qiymatlarni keyingi **so‘rovning parametrlariga bog‘laydi**.

Ikki yangi `SHELL_ERROR` va `SHELL_EXIT_CODE` o‘zgaruvchilari **oxirgi OS buyrug‘i qanday bajarilganligini ko‘rsatadi**.

`\watch` buyrug‘i endi **takrorlashlar sonini** ko‘rsatish imkonini beradi.

postgres_fdw

Tashqi jadvalga **ommaviy kiritish rejimi** endilikda `COPY FROM` buyrug‘i uchun va bo‘lim kalitini yangilashda ham ishlatiladi.

Uchinchi tomon jadvallaridan statistik ma’lumotlarni yig‘ishda **saralashlarni cheklash** uchun `TABLESAMPLE`dan foydalaniladi.

Masofaviy serverlarda ochilgan tranzaktsiyalar endi ketmaket emas, balki **parallel ravishda bekor qilinishi** mumkin.

Endi uchinchi tomon jadvallari uchun **`TRUNCATE` triggerlarini** yaratish mumkin.

pg_dump

`pg_dump` yordamchi dasturi endi **LZ4 va zstd siqish algoritmlarini** qo‘llab-quvvatlaydi.

Endi **bo‘lingan jadvalni barcha bo‘limlarini ularning nomlarini aniq ro‘yxatga kiritmasdan yuklash mumkin**.

Bir nechta jadvallarni yuklash optimallashtirildi: ular birma-bir emas, bitta `LOCK TABLE` buyrug‘i bilan bloklanadi.

pg_waldump va pg_walinspect

`pg_walinspect` kengaytmasi va `pg_waldump` utilita dasturi **sahifalarning to'liq tasvirlari** haqida ma'lumot olishni o'rgandi.

`pg_get_wal_stats`, `pg_get_wal_block_info`, `pg_get_wal_records_info` funksiyalari endi `end_lsn` parametri ko'rsatilmagan bo'lsa, **jurnalni oxirigacha o'qiydi**.

Konfiguratsiya parametrlari

Server konfiguratsiyasining infratuzilmasi parametr qiymatlariga, jumladan, **foydalanuvchi sozlamalariga kirishni tezlashtirish** uchun optimallashtirilgan.

`initdb` utilita dasturini chaqirishda siz **istalgan parametrning qiymatini bekor qilishingiz mumkin**. Bu klaster ishga tushirilgandan so'ng konfiguratsiya fayllarini alohida tahrirlashdan qochadi.

Mahalliylashtirish

lc_collate va *lc_ctype* opsiyalari **olib tashlandi**, chunki ular faqat libc uchun tegishli edi. Provayderni `pg_database`da ko'rish kerak.

ICU provayderi uchun **Unicode tartiblash qoidasi qo'shildi**, u sukut holatda oldindan yozilgan **UNICODE** algoritmini almashtiradi.

Lokal nomlar BCP 47 qoidalariga (en.wikipedia.org/wiki/IETF_language_tag) muvofiq **qisqartiriladi**, bu noto'g'ri tartiblash

qoidasini tanlash imkoniyatini yoʻq qiladi. Va yangi buyruq parametri `CREATE COLLATION` – **oʻz qoidalarini sozlash** imkonini beradi.

Turli xil

Meson tizimi hozirdanoq Windows uchun PostgreSQLni yigʻishni soddalashtirayabdi va tezlashtirayabdi, vaqt oʻtishi bilan Unixga oʻxshash tizimlar uchun u odatiy boʻlgan avtokonf va makeni almashtiradi.

Toʻgridan-toʻgʻri kiritish-chiqarish — MBBT dasturchilarning ertami kech duch keladigan zaruratidir. *io_direct* parametrini yoqish hozircha ishni sekinlashtiradi, chunki buferlangan kiritish/chiqarish holatida OT qabul qiladigan oldindan yuklash hali amalga oshirilmagan. Ammo bu muhim ish boshlandi.

`REFRESH MATERIALIZED VIEW` buyrugʻini `CONCURRENTLY` parametri yordamida chaqirib koʻrinishlarni yangilashda **predikat bloklashlari qoʻshildi**.

Kengaytmalar oʻzlarining WAL resurs menejerlarini yaratishi va shunga mos ravishda oʻz formatida qayd yozuvlarini yaratishi mumkin. Foydalanuvchi nuqtai nazaridan, hech narsa oʻzgarmadi, lekin yangi foydalanish usullarining paydo boʻlishi tomon yana bir muhim qadam qoʻyildi.

Hujjatlar

Hujjatlarga tranzaktsiyalar boʻyicha yangi bob qoʻshildi: postgrespro.ru/docs/postgresql/16/transactions.

28 ii

Ingliz tilidagi hujjatlarning boʻlimlariga havola qilish osonlashdi: kursorni olib borganingizda, nusxa koʻchirishingiz mumkin boʻlgan URL manzili paydo boʻladi. Rus tilidagi tarjimada shunga oʻxshash xususiyat amalga oshirilganiga ancha vaqt boʻldi.

III O‘rnatish va ishlatishni boshlash

PostgreSQL bilan ishlashni boshlash uchun nima kerak? Ushbu bobda biz PostgreSQL xizmatini qanday o‘rnatish va boshqarishni tushuntiramiz, keyingi bobda esa oddiy ma‘lumotlar bazasini yaratamiz va undan so‘rovlarni shakllantirishda foydalaniladigan SQL tili asoslarini o‘rgatish uchun foydalanamiz.

Biz PostgreSQL 16ning odatiy (“vanilli”) distributivini olamiz. PostgreSQL serveri sizda qanday operatsion tizimiligiga qarab har xil tarzda o‘rnatiladi va ishga tushiriladi:

- agar Windows bo‘lsa, o‘qishda davom eting;
- agar Ubuntu – yoki Debian oilasidagi Linux bo‘lsa — 34 betga o‘ting.

Boshqa operatsion tizimlar uchun o‘rnatish bo‘yicha ko‘rsatmalar bu yerda mavjud: postgresql.org/download.

Xuddi shu tarzda siz Postgres Pro Standard 16distibutividan foydalanishingiz mumkin: u oddiy PostgreSQL MBBT bilan to‘liq mos keladi, shuningdek, Postgres Professional kompaniyasida ishlab chiqilgan ba‘zi ishlanmalarni o‘z ichiga oladi va axborot va ta‘lim maqsadlarida foydalanilganda bepul. Uni tanlaganingizda, postgrespro.ru/products/download veb-saytida o‘rnatish bo‘yicha ko‘rsatmalarga qarang.

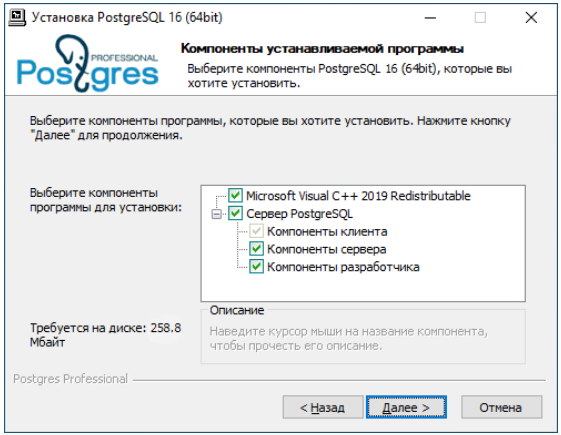
Windows

O'rnatish

O'rnatuvchini yuklab oling (postgrespro.ru/windows), uni ishga tushiring va o'rnatish tilini tanlang.

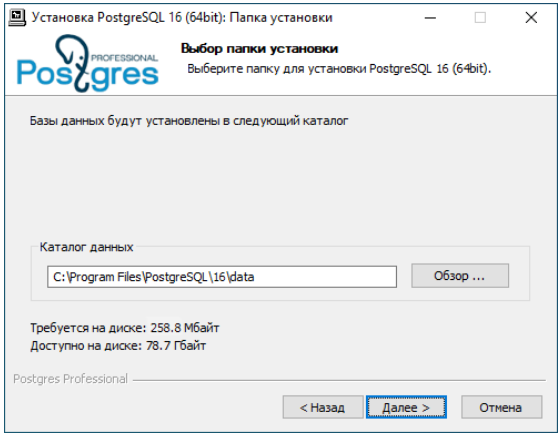
O'rnatuvchi an'anaviy "Мастер" uslubida qurilgan: agar siz hamma narsadan mamnun bo'lsangiz, "Далее" tugmasini bosing. Keling, asosiy o'rnatish bosqichlarini ko'rib chiqamiz.

O'rnatiladigan dasturning tarkibiy qismlari (agar tanlovga ikkilansangiz, hech narsani o'zgartirmang):



Keyin PostgreSQLni o'rnatish uchun katalogni tanlash keladi. O'zgarishsiz, o'rnatish C:\Program Files\PostgreSQL\16 jildida amalga oshiriladi.

Ma'lumotlar bazalari uchun katalog manzilini alohida tanlashingiz mumkin:

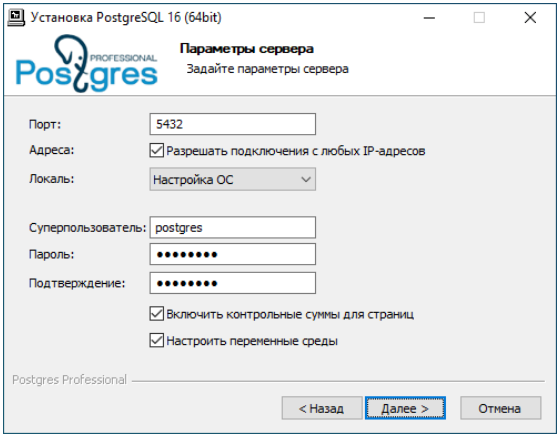


Aynan shu yerda MBBTda saqlanadigan ma'lumotlar joylashadi, shuning uchun agar siz ko'p ma'lumotlarni saqlashni rejalashtirmoqchi bo'lsangiz, diskda yetarli joy mavjudligiga ishonch hosil qiling.

Ingliz tilidan boshqa tildagi ma'lumotlar bilan ishlash uchun tegishli tilni tanlang (masalan, "Russian, Russia") yoki Windowsda kerakli til o'rnatilgan bo'lsa, "Настройка ОС" variantini qoldiring.

MBBT `postgres` foydalanuvchisi parolini kiriting (va qayta kiritish orqali tasdiqlang). Shuningdek, joriy OT hisobi ostida PostgreSQL serveriga ulanish uchun "Настроить переменные среды" katagiga belgi qo'ying.

Qolgan parametrlar o‘zgarishsiz qoldirilishi mumkin:

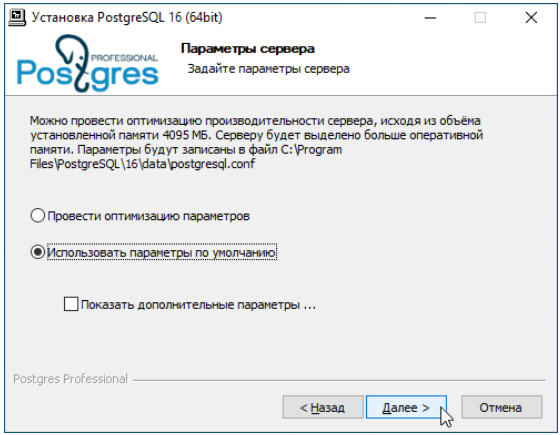


Agar siz PostgreSQLni faqat tanishish uchun o‘rnatishni rejalashtirayotgan bo‘lsangiz, keyingi oynada MBBT ko‘p RAMni egallamasligi uchun “Использовать параметры по умолчанию” xususiyatni belgilang.

Xizmatlarni boshqarish va asosiy fayllar

PostgreSQL o‘rnatilgach operatsion tizimda “postgresql-16” xizmati ro‘yxatga olinadi. Kompyuter Network Service (tarmoq xizmati) hisob qaydnomasi ostida ishga tushganda u ham avtomat ravishda ishga tushadi. Agar kerak bo‘lsa, xizmat parametrlarini standart Windows vositalari yordamida o‘zgartirish mumkin.

Ma'lumotlar bazasi serveri xizmatini vaqtincha to'xtatish uchun o'rnatish vaqtida ko'rsatilgan "Пуск" menyusi jildidan "Stop Server" dasturini ishga tushiring.



Xizmat huddi shunday u yerda joylashgan "Start Server" dasturi tomonidan ishga tushiriladi.

Agar xizmatni ishga tushirishda xatolik yuzaga kelsa, bu haqda server xabarlari jurnalida qayd qilinadi. Jurnal, ma'lumotlar bazalari uchun, o'rnatish paytida tanlangan katalogning log qismkatalogida joylashgan (odatda C:\Program Files\PostgreSQL\16\data\log) bo'ladi. Jurnal shunday tuzilganki, yozuv vaqti-vaqti bilan yangi faylga o'tadi. Joriy faylni – oxirgi o'zgartirish sanasi yoki o'tish sanasini hamda vaqtini o'z ichiga olgan nom bilan topishingiz mumkin.

Server sozlamalarini belgilaydigan bir nechta muhim konfiguratsiya fayllari mavjud. Ular ma'lumotlar bazasi katalogida joylashgan. PostgreSQL bilan birinchi marta tanishayotganda ularni o'zgartirishning hojati yo'q, lekin haqiqiy ishda ular sizga albatta kerak bo'ladi. Ushbu fayllarni ko'rib chiqing – ular yaxshi hujjatlashtirilgan:

- postgresql.conf – server parametrlarining qiymatlarini o'z ichiga olgan asosiy konfiguratsiya fayli;
- pg_hba.conf – kirish huquqi sozlamalarini belgilaydigan fayl. Xavfsizlik maqsadida, sukut bo'yicha, kirish parol bilan tasdiqlanishi kerak va faqat mahalliy kompyuterdangina shunday qilish uchun ruxsat etiladi.

Endi biz ma'lumotlar bazasiga ulanishga tayyormiz. Buyruqlar berish va so'rovlarni bajarishga harakat qilamiz. 39 betdagi "SQLni sinab ko'ramiz" bo'limiga o'ting.

Debian va Ubuntu

O'rnatish

Agar siz Linuxdan foydalanayotgan bo'lsangiz, o'rnatish uchun siz PGDG (PostgreSQL Global Development Group) paketlar bazasini ulashingiz kerak. Hozirda Debian uchun 10 "Buster", 11 "Bullseye" va 12 "Bookworm" talqinlar, Ubuntu uchun esa – 20.04 "Focal", 22.04 "Jammy", 23.04 "Lunar" va 23.10 "Mantic" talqinlar qo'llab-quvvatlanadi.

Terminalda quyidagi buyruqlarni bajaring:

```
$ sudo apt-get install lsb-release
```

```
$ sudo sh -c 'echo "deb \
  http://apt.postgresql.org/pub/repos/apt/ \
  $(lsb_release -cs)-pgdg main" \
  > /etc/apt/sources.list.d/pgdg.list'
```

```
$ wget --quiet -O - \
  https://postgresql.org/media/keys/ACCC4CF8.asc \
  | sudo apt-key add -
```

Ombor linux tizimiga ulandi, endi paketlar ro'yxatini yangilaymiz:

```
$ sudo apt-get update
```

O'rnatishdan oldin mahalliylashtirish sozlamalarini tekshiring

```
$ locale
```

Ingliz tilidan boshqa tildagi ma'lumotlar bilan ishlash uchun LC_CTYPE va LC_COLLATE o'zgaruvchilari qiymatini o'zgartirishingiz kerak bo'lishi mumkin. "en_US.UTF8" tili rus tili uchun ham mos keladi, ammo uni o'zgartirish yaxshidir:

```
$ export LC_CTYPE=ru_RU.UTF8
```

```
$ export LC_COLLATE=ru_RU.UTF8
```

Shuningdek, operatsion tizimda ham tegishli til o'rnatilganligiga ishonch hosil qiling:

```
$ locale -a | grep ru_RU
```

```
ru_RU.utf8
```

Agar bunday bo'lmasa, uni generatsiya qilib yarating:

```
$ sudo locale-gen ru_RU.utf8
```

Endi siz o'rnatishni boshlashingiz mumkin:

```
$ sudo apt-get install postgresql-16
```

Bu oxirgi bosqich edi. Endi PostgreSQL MBBT o'rnatildi, ishga tushirildi va foydalanishga tayyor. Buni tekshirish uchun quyidagi buyruqni bering:

```
$ sudo -u postgres psql -c 'select version()'
```

Agar hamma narsa muvaffaqiyatli bajarilgan bo'lsa, javob sifatida PostgreSQL talqinini olishingiz kerak.

Xizmatlarni boshqarish va asosiy fayllar

O'rnatish vaqtida PostgreSQL maxsus `postgres` hisobini yaratadi, uning ostida serverga xizmat ko'rsatish protseslari ishlaydi va MBBT bilan bog'liq barcha fayllarga egalik qiladi. Operatsion tizim qayta ishga tushirilganda PostgreSQL avtomatik ravishda ishga tushadi. Standart sozlamalar bilan bu muammo emas, chunki serverga murojaatlar bo'lmasa, tizim resurslari juda kam sarflanadi. Agar siz baribir avtomatik ishga tushirishni o'chirib qo'ymoqchi bo'lsangiz, quyidagilarni bajaring:

```
$ sudo systemctl disable postgresql
```

Ma'lumotlar bazasi serveri xizmatini vaqtincha to'xtatish uchun quyidagi buyruqni bering:

```
$ sudo systemctl stop postgresql
```

Server xizmatini quyidagi buyruq bilan ishga tushirishingiz mumkin:

```
$ sudo systemctl start postgresql
```

Shuningdek, joriy holatini tekshirishingiz ham mumkin:

```
$ sudo systemctl status postgresql
```

Agar xizmat ishga tushmasa, server xabarlari jurnali sababni topishga yordam beradi. /var/log/postgresql/postgresql-16-main.log fayldagi eng so'nggi jurnal yozuvlarini diqqat bilan o'qing.

Ma'lumotlar bazasidagi barcha ma'lumotlar fayl tizimida /var/lib/postgresql/16/main/ maxsus katalogida joylashgan. Agar siz, juda ko'p ma'lumotlarni saqlamoqchi bo'lsangiz, uning uchun yetarli joy mavjudligiga – ishonch hosil qiling.

Server sozlamalarini belgilaydigan bir nechta muhim konfiguratsiya fayllari mavjud. Ishni boshlashda ushbu fayllarni o'zgartirishning hojati yo'q, lekin ular bilan oldindan tanishgan ma'qul – kelajakda sizga albatta kerak bo'ladi:

- /etc/postgresql/16/main/postgresql.conf – server parametrlari qiymatlarini o'z ichiga olgan asosiy konfiguratsiya fayli;
- /etc/postgresql/16/main/pg_hba.conf – kirish huquqi sozlamalarini belgilaydigan fayl. Xavfsizlik maqsadida, sukut bo'yicha, kirish faqat mahalliy kompyuterdan va faqat nomi operatsion tizim hisobi nomiga mos keladigan ma'lumotlar bazasi foydalanuvchisi nomidan ruxsat etiladi.

Ma'lumotlar bazasiga ulanish va SQLni amalda sinab ko'rish vaqti keldi.

IV SQLni sinab koʻramiz

Psql yordamida ulanish

MBBT serveriga ulanish va har qanday buyruqlarni bajarish uchun mijoz dasturi talab qilinadi. “Ilova uchun PostgreSQL” boʻlimida biz turli xil dasturlash tillarida dasturlardan soʻrovlarni qanday yuborish haqida gapiramiz, ammo endi biz buyruq qatori rejimida interaktiv ishlash mumkin boʻlgan psql terminal mijozi haqida gapiramiz.

Afsuski, bugungi kunda koʻpchilik buyruq qatorini yoqtirmaydi. Nima uchun u bilan ishlashni oʻrganish maqsadga muvofiq deb qaraladi?

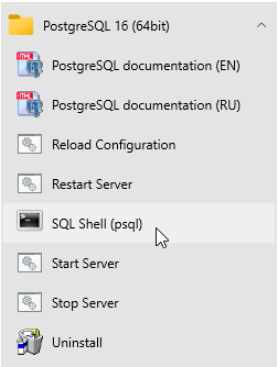
Birinchidan, `psql` – standart mijoz boʻlib, u har qanday PostgreSQL yigʻmasiga kiritilgan va shuning uchun har doim qoʻlingizda. Sizga moslashtirilgan muhitga ega boʻlish, albatta, yaxshi, lekin notanish muhitda ojiz boʻlish mantiqqa toʻgʻri kelmaydi.

Ikkinchidan, `psql` maʻlumotlar bazasini administratorning kundalik vazifalarini hal qilish, kichik soʻrovlarni yozish va jarayonlarni avtomatlashtirish uchun juda qulaydir, masalan, vaqti-vaqti bilan MBBT serverida dastur kodini yangilashni oʻrnatish. Uning oʻz buyruqlari mavjud boʻlib, ular maʻlumotlar bazasida saqlangan obyektlar boʻylab harakatlanish va jadvallardagi maʻlumotlarni vizual shaklda taqdim etish imkonini beradi.

Agar siz grafik foydalanuvchi interfeyslari bilan ishlashga odatlangan bo‘lsangiz, pgAdminni sinab ko‘ring – bu haqda keyinroq aytib o‘tamiz – yoki boshqa shunga o‘xshash mahsulotlarni quyidagi manzildan bilib olishingiz mumkin: wiki.postgresql.org/wiki/Community_Guide_to_PostgreSQL_GUI_Tools

Linux operatsion tizimida `psql`-ni ishga tushirish uchun quyidagi buyruqni bering:

```
$ sudo -u postgres psql
```



Windows operatsion tizimida “Пуск” menyusidan “SQL Shell (psql)” dasturini ishga tushiring. Parol so‘ralganda, PostgreSQLni o‘rnatishda ko‘rsatilgan `postgres` foydalanuvchi parolini kiriting.

Windows foydalanuvchilari terminalda kirillcha belgilar to‘g‘ri ko‘rsatilmasligi bilan bog‘liq muammolarga duch kelishlari mumkin. Bunday holda, terminal oynasi xususiyatlarida TrueType shrift o‘rnatilganligiga ishonch hosil qiling (odatda “Lucida Console” yoki “Consolas”).

Shunday qilib, taklif ikkala operatsion tizimda ham bir xil ko‘rinadi: `postgres=#` (“postgres” bu yerda siz hozir ulangan ma‘lumotlar bazasining nomi). Bitta server bir nechta

ma‘lumotlar bazasiga xizmat qilishi mumkin, lekin u bir vaqtning o‘zida faqat bittasi bilan ishlashi mumkin.

Endi birlamchi buyruqlarni o‘rganamiz. Faqat qalin harf bilan yozilganlarni kiriting. Tizimning taklifi va buyruqlarga javoblari qulaylik uchun taqdim etiladi.

Ma‘lumotlar bazasi

Test nomli yangi ma‘lumotlar bazasini yaratamiz. Quyidagilarni bajaring:

```
postgres=# CREATE DATABASE test;

CREATE DATABASE
```

Buyruqning oxiridagi nuqtali vergulni unutmang – PostgreSQL ushbu belgini ko‘rmaguncha, u siz yozishni davom ettirayotganingizni taxmin qiladi (ya‘ni, buyruq bir nechta qatorlarga bo‘linishi mumkin).

Endi yaratilgan ma‘lumotlar bazasiga o‘tamiz:

```
postgres=# \c test

You are now connected to database "test" as user
"postgres".

test=#
```

Ko‘rib turganingizdek, taklif `test=#`ga o‘zgartirildi.

Biz kiritgan buyruq SQLga o‘xshamaydi – u teskari chiziq bilan boshlanadi. Faqat `psql` tushunadigan maxsus buyruqlar shunday ko‘rinadi (shuning uchun agar sizda pgAdmin yoki boshqa grafik vosita ochiq turgan bo‘lsa, slash bilan boshlangan hamma narsani o‘tkazib yuboring).

Bir nechta `psql` buyruqlari mavjud va biz ularning ba'zilari bilan biroz keyinroq tanishamiz, ammo to'liq ro'yxatni qisqacha tavsiflari bilan hoziroq tanishib olishingiz mumkin:

```
test=# \?
```

Yordam ma'lumotlari juda ko'p bo'lgani uchun u operatsion tizimda sozlangan peyjer-buyrug'i yordamida ko'rsatiladi (odatda `more` yoki `less`).

Jadvallar

Relyatsion MBBTlarda ma'lumotlar **jadvallar** ko'rinishida taqdim etiladi. Jadvalning tuzilishi uning **ustunlari** bilan belgilanadi. Haqiqiy ma'lumotlar **qatorlarda** joylashgan. Ular tartibsiz saqlanadi va hatto jadvalga qo'shilgan tartibda bo'lishi shart emas.

Har bir ustun ma'lumotlar turiga ega. Qatorlardagi maydon qiymatlari ushbu turlarga mos kelishi kerak. PostgreSQL ko'p sonli o'z ichida o'rnatilgan turlarga (postgrespro.ru/doc/datatype) va yangilarini yaratish imkoniyatlariga ega, ammo biz o'zimizni eng asosiylari bilan cheklaymiz:

- integer – butun sonlar;
- text – matnli qatorlar;
- boolean – mantiqiy tur, `true` ("rost") yoki `false` ("yolg'on") qiymat qabul qila oladi.

Ma'lumotlar turi bilan belgilangan normal qiymatlarga qo'shimcha ravishda, maydon **aniqlanmagan qiymat ma'nosidagi** NULL qiymatiga ega bo'lishi mumkin – bu "noma'lum qiymat" yoki "qiymat o'rnatilmagan" deb hisoblanishi mumkin.

Keling universitetda o'qitiladigan fanlar jadvalini tuzamiz:

```
test=# CREATE TABLE courses(
test(#   c_no text PRIMARY KEY,
test(#   title text,
test(#   hours integer
test(# );

CREATE TABLE
```

`Psql` taklifi qanday o'zgarganiga e'tibor bering: bu buyruq yangi satrda davom etishiga ishora. Kelajakda qulaylik uchun biz takliflarni har bir satrda takrorlamaymiz.

Ushbu buyruq yordamida `courses` nomli jadval uchta ustundan iborat bo'lishini aniqladik: `c_no` – kursning matnli raqami, `title` – kurs nomi va `hours` – ma'ruza soatlarining butun soni.

Ustunlar va ma'lumotlar turlari bilan birgalikda, siz avtomatik ravishda tekshiriladigan yaxlitlik cheklovlarini kiritishingiz mumkin, bu cheklovlar yordamida – MBBT ma'lumotlar bazasida noto'g'ri ma'lumotlar paydo bo'lishiga yo'l qo'ymaydi. Bizning misolimiz `c_no` ustuniga `PRIMARY KEY` cheklovini qo'shadi; endi u takroriy yoki aniqlanmagan qiymatlarga ruxsat bermaydi. Bunday ustun yordamida siz qatorlarni bir-biridan farqlashingiz mumkin. Butunlik cheklovlarining to'liq ro'yxati postgrespro.ru/doc/ddl-constraints sahifasida mavjud.

`CREATE TABLE` buyrug'ining aniq sintaksisini hujjatlarda yoki to'g'ridan-to'g'ri `psql`da topish mumkin:

```
test=# \help CREATE TABLE
```

Har bir SQL buyrug'i uchun shunday yordam ma'lumoti mavjud. Buyruqlarning to'liq ro'yxatini `\help` buyrug'ini parametrlarsiz chaqirib ko'riladi.

Jadvallarni to‘ldirish

Yaratilgan jadvalga bir necha qator qo‘shamiz:

```
test=# INSERT INTO courses(c_no, title, hours)
VALUES ('CS301', 'Базы данных', 30),
       ('CS305', 'Сети ЭВМ', 60);
```

```
INSERT 0 2
```

INSERT buyrug‘i tashqi manbadan ma‘lumotlarni ommaviy yuklash uchun mos emas, lekin buning uchun maxsus ishlab chiqilgan COPY buyrug‘i mavjud: postgrespro.ru/doc/sql-copy.

Keling, ma‘lumotlar bazasida yana ikkita jadval yarataylik: “Talabalar” va “Imtihonlar”. Har bir talaba uchun uning ismi va qabul yili saqlanib qolsin va u talabalik ID raqami bilan aniqlanadigan qilinsin deb talab qo‘yaylik.

```
test=# CREATE TABLE students(
  s_id integer PRIMARY KEY,
  name text,
  start_year integer
);
```

```
CREATE TABLE
```

```
test=# INSERT INTO students(s_id, name, start_year)
VALUES (1451, 'Анна', 2014),
       (1432, 'Виктор', 2014),
       (1556, 'Нина', 2015);
```

```
INSERT 0 3
```

Imtihon jadvalida talabalarning turli fanlar bo‘yicha olgan baholari to‘g‘risidagi ma‘lumotlar mavjud. Shunday qilib, talabalar va fanlar bir-biri bilan ko‘p-ko‘p munosabatda bog‘langan: bir talaba ko‘p fanlardan imtihon topshirishi mumkin va ko‘plab talabalar bir fandan imtihon topshirishlari mumkin.

Imtihon jadvalidagi yozuv talaba kartasi raqami va kurs raqami kombinatsiyasi bilan aniqlanadi. Bir vaqtning o'zida bir nechta ustunlarga qo'llaniladigan bu yaxlitlik cheklovi CONSTRAINT bandi yordamida aniqlanadi:

```
test=# CREATE TABLE exams(
  s_id integer REFERENCES students(s_id),
  c_no text REFERENCES courses(c_no),
  score integer,
  CONSTRAINT pk PRIMARY KEY(s_id, c_no)
);

CREATE TABLE
```

Bundan tashqari, REFERENCES bandidan foydalanib, biz **tashqi kalitlar** deb ataladigan ikkita havolali yaxlitlik cheklovlarini qo'shdik. Bunday cheklovlar bir jadvaldagi qiymatlar boshqa jadvaldagi qatorlarga **havola** ekanligini ko'rsatadi.

Endi har qanday harakat uchun MBBT imtihon jadvalida ko'rsatilgan barcha s_id identifikatorlarining haqiqiy talabalarga (ya'ni talabalar jadvalidagi yozuvlar), shuningdek c_no – raqamining haqiqiy kurslarga muvofiqligini tekshiradi. Shunday qilib, mavjud bo'lmagan talabaga yoki mavjud bo'lmagan fanga baho qo'yish imkoniyati istisno qilinadi – foydalanuvchining harakatlari yoki ilovadagi mumkin bo'lgan xatolardan qat'i nazar. Keling, o'quvchilarimizga bir nechta baholarni beramiz:

```
test=# INSERT INTO exams(s_id, c_no, score)
VALUES (1451, 'CS301', 5),
       (1556, 'CS301', 5),
       (1451, 'CS305', 5),
       (1432, 'CS305', 4);

INSERT 0 4
```

Ma‘lumotlarni olish

Oddiy so‘rovlar

Jadvallardan ma‘lumotlarni o‘qish SQL SELECT operatori yordamida amalga oshiriladi. Masalan, courses jadvalidan faqat ikkita ustunni ko‘rsatamiz. AS konstruksiyasi, agar kerak bo‘lsa, ustun nomini o‘zgartirishga imkon beradi:

```
test=# SELECT title AS course_title, hours
FROM courses;

 course_title | hours
--------------+-------
 Базы данных  |    30
 Сети ЭВМ     |    60
(2 rows)
```

Barcha ustunlarni ma‘lumotlarini olish uchun yulduzcha belgisini ko‘rsatish kifoya:

```
test=# SELECT * FROM courses;

 c_no  |    title    | hours
-------+-------------+-------
 CS301 | Базы данных |    30
 CS305 | Сети ЭВМ    |    60
(2 rows)
```

Ishlab chiqarish kodida so‘rov yanada samarali bajarilishi va natija yangi ustunlar paydo bo‘lishiga bog‘liq bo‘lmasligi uchun faqat kerakli ustunlarni aniq ko‘rsatish yaxshiroqdir. Ammo interaktiv so‘rovlar uchun “yulduzcha” juda qulay.

So‘rov bo‘yicha chiqish bir xil qatorlarni o‘z ichiga olishi mumkin. Barcha ustunlar – ko‘rsatilmaganda, ular asl jadvalda bo‘lmagan taqdirda ham dublikatlar (takrorlanuvchilar) paydo bo‘lishi mumkin:

```
test=# SELECT start_year FROM students;

 start_year
------------
       2014
       2014
       2015
(3 rows)
```

Qabul qilish yilining barcha **takrorlanmas** qiymatlarini tanlash uchun SELECTdan keyin DISTINCT so'zini qo'shishingiz kerak:

```
test=# SELECT DISTINCT start_year FROM students;

 start_year
------------
       2014
       2015
(2 rows)
```

Qo'shimcha ma'lumot olish uchun ushbu havolaga qarang: postgrespro.ru/doc/sql-select#SQL-DISTINCT

Umuman olganda, SELECT so'zidan keyin istalgan iboralarni ko'rsatishingiz mumkin va FROM bandisiz so'rov bitta qatorni qaytaradi. Masalan:

```
test=# SELECT 2+2 AS result;

 result
--------
      4
(1 row)
```

Odatda, ma'lumotlarni olishda siz barcha qatorlarni emas, balki faqat ba'zi shartlarni qondiradigan qatorlarnigina olishni xohlaysiz. Bunday holatlarda filtrlash sharti WHERE bandida yoziladi:

```
test=# SELECT * FROM courses WHERE hours > 45;

 c_no  |  title   | hours
-------+----------+-------
 CS305 | Сети ЭВМ |    60
(1 row)
```

Shart mantiqiy turda bo'lishi kerak. Masalan, =, <> (yoki !=), >, >=, <, <=, operatorlarini o'z ichiga olishi mumkin, shuningdek oddiy shartlarda bo'lgani kabi AND, OR, NOT mantiqiy operatorlari va dasturlash tillaridagidek qavslar yordamida oddiyroq shartlarni birlashmasidan xosil bo'lishi ham mumkin.

Nozik masala – aniqlanmagan NULL qiymatidir. Tanlov faqat filtrlash sharti rost bo'lgan qatorlarni o'z ichiga oladi. Agar qiymat noto'g'ri yoki **aniqlanmagan bo'lsa**, qator o'chiriladi.

Esda tuting:

- biror narsani aniqlanmagan qiymatga solishtirish natijasi aniqlanmagan;
- qiymati aniqlanmagan mantiqiy amallarning natijasi, qoida tariqasida, aniqlanmagan (istisnolar: `true` OR NULL = `true`, `false` AND NULL = `false`);
- qiymatning aniqligini tekshirish uchun IS NULL (IS NOT NULL) va IS DISTINCT FROM (IS NOT DISTINCT FROM) maxsus operatorlari qo'llaniladi.

Nulllar bilan ishlashda `coalesce` ifodasi ("koueles" deb talaffuz qilinadi) ko'pincha NULLni boshqa narsa bilan almashtirish uchun ishlatiladi, masalan, matn turlari uchun bo'sh qator yoki sonli turlar uchun nol.

Qo'shimcha ma'lumot olish uchun havolaga qarang: postgrespro.ru/doc/functions-comparison.

Yaxshi ishlab chiqilgan relyatsion ma‘lumotlar bazasida ortiqcha ma‘lumotlar mavjud emas. Misol uchun, imtihon jadvalida talabaning ismi bo‘lmasligi kerak, chunki uni boshqa jadvalda talaba ID raqami orqali topish mumkin.

Shuning uchun, so‘rovda barcha kerakli qiymatlarni olish uchun siz ko‘pincha FROM bandida ularning nomlarini ko‘rsatib, bir nechta jadvallardagi ma‘lumotlarni birlashtirishingiz kerak:

```
test=# SELECT * FROM courses, exams;

 c_no  |    title    | hours | s_id | c_no  | score
-------+-------------+-------+------+-------+-------
 CS301 | Базы данных |    30 | 1451 | CS301 |     5
 CS305 | Сети ЭВМ    |    60 | 1451 | CS301 |     5
 CS301 | Базы данных |    30 | 1556 | CS301 |     5
 CS305 | Сети ЭВМ    |    60 | 1556 | CS301 |     5
 CS301 | Базы данных |    30 | 1451 | CS305 |     5
 CS305 | Сети ЭВМ    |    60 | 1451 | CS305 |     5
 CS301 | Базы данных |    30 | 1432 | CS305 |     4
 CS305 | Сети ЭВМ    |    60 | 1432 | CS305 |     4
(8 rows)
```

Biz olgan narsa jadvallarning to‘g‘ridan – to‘g‘ri yoki dekart ko‘paytmasi deb ataladi – bitta jadvalning har bir qatori boshqasining har bir qatoriga qo‘shiladi.

Odatda, WHERE bandida birlashish shartini ko‘rsatib, yanada foydali va mazmunli natijalarga erishishingiz mumkin. Kurslarni ushbu kurs uchun maxsus o‘tkazilgan imtihonlar bilan taqqoslab, barcha fanlar bo‘yicha baholarni so‘raymiz:

```
test=# SELECT courses.title, exams.s_id, exams.score
FROM courses, exams
WHERE courses.c_no = exams.c_no;
```

```
     title     | s_id | score
---------------+------+-------
 Базы данных | 1451 |     5
 Базы данных | 1556 |     5
 Сети ЭВМ    | 1451 |     5
 Сети ЭВМ    | 1432 |     4
(4 rows)
```

So'rovlarni JOIN kalit so'zi yordamida birlashmalarni belgilash orqali boshqa shaklda shakllantirish mumkin. "Kompyuter tarmoqlari" kursi bo'yicha talabalar va ularning baholarini ko'rsatamiz:

```
test=# SELECT students.name, exams.score
FROM students
JOIN exams
  ON students.s_id = exams.s_id
  AND exams.c_no = 'CS305';
```

```
  name  | score
--------+-------
 Анна   |     5
 Виктор |     4
(2 rows)
```

MBBT nuqtai nazaridan, bu shakllar bir-biriga ekvivalentdir, shuning uchun siz vizualroq ko'rinadigan usuldan foydalanishingiz mumkin.

Ko'rib turganingizdek, natijada JOIN so'zining chap tomonida ko'rsatilgan jadval qatorlari mavjud emas, ular uchun jadvalda ushbu so'zning o'ng tomonida ko'rsatilgan juftlik shartlarga mos ma'lumotlar yo'q: shart imtihonlarga qo'yiladi, lekin talabalar, imtihondan o'tmaganlar ham bu shartni qanoatlantirmaydi. Namunaga barcha talabalarni kiritish uchun siz tashqi birlashmani ishlatishingiz kerak:

```
test=# SELECT students.name, exams.score
FROM students
LEFT JOIN exams
  ON students.s_id = exams.s_id
  AND exams.c_no = 'CS305';

  name  | score
--------+-------
 Анна   |     5
 Виктор |     4
 Нина   |
(3 rows)
```

Endi natijalar chap jadvaldagi qatorlarni ham o'z ichiga oladi (shuning uchun operatsiya `LEFT JOIN` deb ataladi) garchi ular o'ng tomondagi juftlik shartini qanoatlantirmasa ham. Bunday holda, o'ng jadvalning ustunlari uchun null qiymatlar qaytariladi.

`WHERE`dan keyingi shartlar birlashtirilgan qatorlarga qo'llaniladi, shuning uchun agar siz fanlar bo'yicha cheklovni qo'shilish shartidan `WHERE` bandiga o'tkazsangiz, Nina tanlovga kiritilmaydi – chunki u uchun `exams.c_no` qiymati aniqlanmagan:

```
test=# SELECT students.name, exams.score
FROM students
LEFT JOIN exams ON students.s_id = exams.s_id
WHERE exams.c_no = 'CS305';

  name  | score
--------+-------
 Анна   |     5
 Виктор |     4
(2 rows)
```

Bog'lanishlardan qo'rqmang. Bu relyatsion MBBTlar uchun odatiy va tabiiy jarayon bo'lib, PostgreSQL uni amalga oshirish uchun samarali mexanizmlarning butun arsenaliga ega.

Ilovada ma'lumotlarni ulamang, bu ishni bajarish uchun ma'lumotlar bazasi serveriga ishoning – u bu ishni ajoyib tarzda bajaradi.

Qo'shimcha ma'lumot olish uchun hujjatlarga qarang: postgrespro.ru/doc/sql-select#SQL-FROM.

Qismso'rovlar

SELECT operatori jadvalni shakllantiradi, jadvalni (biz avval ko'rganimizdek) natijada ko'rsatish mumkin yoki jadval mazmunli joylashishi mumkin bo'lgan har qanday joyda boshqa SQL konstruktsiyasida foydalanish mumkin. Qavslar ichiga olingan ushbu ichki o'rnatilgan SELECT operatori **qismso'rov** deb ataladi.

Agar simso'rov aynan bitta satr va bitta ustunni qaytarsa, uni oddiy skalyar ifoda kabi ishlatish mumkin:

```
test=# SELECT name,
  (SELECT score
   FROM exams
   WHERE exams.s_id = students.s_id
   AND exams.c_no = 'CS305')
FROM students;
```

```
  name  | score
--------+-------
 Анна   |     5
 Виктор |     4
 Нина   |
(3 rows)
```

Agar SELECT ifodalari ro'yxatida qo'llanilgan skalyar qismso'rovda qatorlar bo'lmasa, aniqlanmagan qiymat qaytariladi

(bizning misolimizdagi oxirgi qatorida bo'lgani kabi). Shuning uchun, skalyar qismso'rovlarni ularni birlashma bilan almashtirish orqali kengaytirish mumkin, lekin har doim tashqi bog'lanish orqali.

Skalyar qismso'rovlar filtrlashda ham ishlatilishi mumkin. Keling, 2014 yildan keyin qabul qilingan talabalar tomonidan topshirilgan barcha imtihonlarni olaylik:

```
test=# SELECT *
FROM exams
WHERE (SELECT start_year
       FROM students
       WHERE students.s_id = exams.s_id) > 2014;

 s_id | c_no  | score
------+-------+-------
 1556 | CS301 |     5
(1 row)
```

SQLda siz ixtiyoriy qatorlar sonini qaytaradigan qismso'rovlar uchun shartlarni shakllantirishingiz mumkin. Buning uchun bir nechta konstruktsiyalar mavjud, ulardan biri, IN munosabati, qiymat qimsso'rov tomonidan qaytarilgan jadvalda mavjud yoki yo'qligini tekshiradi.

Ko'rsatilgan kursda har qanday baho olgan talabalarni ko'rsatamiz:

```
test=# SELECT name, start_year
FROM students
WHERE s_id IN (SELECT s_id
               FROM exams
               WHERE c_no = 'CS305');

  name  | start_year
--------+------------
 Анна   |       2014
 Виктор |       2014
(2 rows)
```

NOT IN operatori teskari natijani qaytaradi. Masalan, birorta ham a'lo baho olmagan talabalar ro'yxati:

```
test=# SELECT name, start_year
FROM students
WHERE s_id NOT IN
      (SELECT s_id FROM exams WHERE score = 5);

  name   | start_year
---------+------------
 Виктор |       2014
(1 row)
```

E'tibor bering, namunaga hech qanday baho olmagan talabalar ham kiradi.

Yana bir imkoniyat – qismso'rov kamida bitta qatorni qaytarganligini tekshiradigan EXISTS predikatidan foydalanish. Undan foydalanib, oldingi so'rovni boshqa shaklda yozishingiz mumkin:

```
test=# SELECT name, start_year
FROM students
WHERE NOT EXISTS (SELECT s_id
                  FROM exams
                  WHERE exams.s_id = students.s_id
                  AND score = 5);

  name   | start_year
---------+------------
 Виктор |       2014
(1 row)
```

Qo'shimcha ma'lumot olish uchun hujjatlarga qarang: postgrespro.ru/doc/functions-subquery.

Yuqorida biz noaniqlikni oldini olish uchun ustun nomlarini jadval nomlari bilan to'ldirganmiz, lekin ba'zida bu yetarli emas. Masalan, so'rovda bir jadval ikki marta ishlatilishi mumkin yoki FROM bandidagi jadval o'rnini nomsiz qismso'rov

egallashi mumkin. Bunday hollarda siz qismsoʻrovdan keyin uning taxallusini koʻrsatishingiz mumkin. Oddiy jadvallarga taxalluslar ham berilishi mumkin.

“Maʻlumotlar bazasi” fanidan oʻquvchilarning ism-shariflari va baholarini koʻrsatamiz:

```
test=# SELECT s.name, ce.score
FROM students s
JOIN (SELECT exams.*
      FROM courses, exams
      WHERE courses.c_no = exams.c_no
      AND courses.title = 'Базы данных') ce
  ON s.s_id = ce.s_id;

 name | score
------+-------
 Анна |     5
 Нина |     5
(2 rows)
```

Bu yerda `s` – jadval taxallus, `ce` – esa qismsoʻrov taxallusidir. Taxalluslarni qisqa, ammo tushunarli qilib qoʻyish yaxshidir.

Xuddi shu soʻrov qismsoʻrovlarsiz yozilishi mumkin:

```
test=# SELECT s.name, e.score
FROM students s, courses c, exams e
WHERE c.c_no = e.c_no
AND c.title = 'Базы данных'
AND s.s_id = e.s_id;
```

Tartiblash

Yuqorida aytib oʻtilganidek, jadvallardagi maʻlumotlar tartiblanmagan. Natija qatorlarini maʻlum bir tartibda koʻrsatish uchun `ORDER BY` bandidan foydalaning. Har bir ifodadan keyin (tartiblash kaliti) yoʻnalishni belgilashingiz mumkin: `ASC` –

o‘sish (bu tartib sukut bo‘yicha ishlatiladi) – yoki DESC – pasayish.

```
test=# SELECT *
FROM exams
ORDER BY score, s_id, c_no DESC;
```

```
 s_id | c_no  | score
------+-------+-------
 1432 | CS305 |     4
 1451 | CS305 |     5
 1451 | CS301 |     5
 1556 | CS301 |     5
(4 rows)
```

Bu erda qatorlar balning os‘ishi bo‘yicha tartiblangan. Agar baholar teng bo‘lsa – talaba ID raqamining o‘sish tartibida olinadi. Agar ikkala birinchi kalit ham mos kelsa – kurs raqamining kamayish tartibida olinadi.

Natijani olishdan oldin, so‘rov oxirida saralash amalini bajarish mantiqan to‘g‘ri keladi, qismso‘rovlarda odatda ma‘nosiz.

Qo‘shimcha ma‘lumot olish uchun hujjatlarga qarang: postgrespro.ru/doc/sql-select#SQL-ORDERBY.

Guruhlash

Guruhlashda manba jadvallarining bir necha qatoridagi ma‘lumotlar asosida hisoblangan qiymat natijaning bir qatoriga joylashtiriladi. Agregat funktsiyalari guruhlash bilan birga ishlatiladi. Masalan, o‘tkazilgan imtihonlarning umumiy sonini, ularni topshirgan talabalar sonini va o‘rtacha ballni ko‘rsatamiz:

```
test=# SELECT count(*), count(DISTINCT s_id),
avg(score)
FROM exams;

 count | count |        avg
-------+-------+--------------------
     4 |     3 | 4.7500000000000000
(1 row)
```

Xuddi shu ma'lumotni guruhlash kalitlarini ko'rsatadigan GROUP BY bandi yordamida kurs raqami bo'yicha guruhlab olish mumkin:

```
test=# SELECT c_no, count(*),
count(DISTINCT s_id), avg(score)
FROM exams
GROUP BY c_no;

 c_no  | count | count |        avg
-------+-------+-------+--------------------
 CS301 |     2 |     2 | 5.0000000000000000
 CS305 |     2 |     2 | 4.5000000000000000
(2 rows)
```

Agregat funktsiyalarning to'liq ro'yxati: postgrespro.ru/doc/functions-aggregate.

Guruhlashdan foydalanganda, siz birlashtirish natijalariga ko'ra qatorlarni filtrlashingiz kerak bo'lishi mumkin. Bunday shartlar HAVING bandida ko'rsatilishi mumkin. WHEREdan farqi shundaki, guruhlashdan oldin WHERE shartlari qo'llaniladi (ular manba jadvallari ustunlaridan foydalanishi mumkin) va guruhlashdan keyin HAVING – shartlari qo'llaniladi (va ular natijalar jadvalidagi ustunlardan ham foydalanishlari mumkin).

Har qanday fandan birdan ortiq 5 baho olgan talabalarning ismlarini chiqaramiz:

```
test=# SELECT students.name
FROM students, exams
WHERE students.s_id = exams.s_id AND exams.score = 5
GROUP BY students.name
HAVING count(*) > 1;

 name
------
 Анна
(1 row)
```

Qo'shimcha ma'lumot uchun: postgrespro.ru/doc/sql-select#SQL-GROUPBY.

Ma'lumotlarni o'zgartirish va o'chirish

Jadvaldagi ma'lumotlarni o'zgartirish WHERE bandi bilan belgilangan satrlar uchun yangi maydon qiymatlarini belgilaydigan UPDATE bayonoti orqali amalga oshiriladi (SELECT iborasidagi kabi).

Masalan, "Ma'lumotlar bazalari" kursi uchun ma'ruza soatlarini ikki baravar oshiramiz:

```
test=# UPDATE courses
SET hours = hours * 2
WHERE c_no = 'CS301';

UPDATE 1
```

Qo'shimcha ma'lumot uchun: postgrespro.ru/doc/sql-update.

DELETE operatori belgilangan jadvaldan WHERE bandida ko'rsatilgan shartlarga mos qatorlarni olib tashlaydi:

```
test=# DELETE FROM exams WHERE score < 5;

DELETE 1
```

Tranzaksiyalar

Keling, ma'lumotlar sxemasini murakkablashtiramiz va o'quvchilarni guruhlarga taqsimlaymiz va har bir guruhning yetakchisi bo'lishni ta'minlaymiz. Buning uchun guruhlar jadvalini yaratamiz:

```
test=# CREATE TABLE groups(
  g_no text PRIMARY KEY,
  monitor integer NOT NULL REFERENCES students(s_id)
);

CREATE TABLE
```

Bu yerda biz null qiymatlarni taqiqlovchi `NOT NULL` yaxlitlik cheklovidan foydalandik.

Endi bizga yana bitta ustun kerak – guruh raqami. Yaxshiyamki, siz mavjud jadvalga yangi ustun qo'shishingiz mumkin:

```
test=# ALTER TABLE students
ADD g_no text REFERENCES groups(g_no);

ALTER TABLE
```

`Psql` buyrug'i yordamida jadvalda qaysi ustunlar aniqlanganligini har doim ko'rishingiz mumkin:

```
test=# \d students

      Table "public.students"
   Column   |  Type   | Modifiers
------------+---------+----------
 s_id       | integer | not null
 name       | text    |
 start_year | integer |
 g_no       | text    |
 ...
```

Ma'lumotlar bazasida qaysi jadvallar mavjudligini ham ko'rishingiz mumkin:

```
test=# \d

          List of relations
 Schema |   Name   | Type  | Owner
--------+----------+-------+----------
 public | courses  | table | postgres

 public | exams    | table | postgres
 public | groups   | table | postgres
 public | students | table | postgres
(4 rows)
```

Keling, "A-101" guruhini yarataylik va unga barcha talabalarni joylashtiramiz va Annani bosh qiz qilaylik.

Bu erda ahamiyatsiz bo'lmagan holat bor. Rahbarni ko'rsatmasdan guruh yarata olmaysiz, lekin siz talabani u a'zo bo'lmagan guruhga yetakchi qilib tayinlay olmaysiz – bu ma'lumotlar bazasida mantiqan noto'g'ri, nomuvofiq ma'lumotlar paydo bo'lishiga olib keladi. Bu ikki amal alohida ma'noga ega emas: ular bir vaqtning o'zida bajarilishi kerak. Mantiqiy jihatdan bo'linmaydigan ish birligini tashkil etuvchi amallar guruhi **tranzaksiya** deyiladi.

Tranzaktsiyani boshlaylik:

```
test=# BEGIN;

BEGIN
```

Keyin boshliq bilan birga guruh qo'shamiz. Biz talabalar raqamlarini yoddan eslab qolishimiz shart emas, shuning uchun biz so'rovni to'g'ridanto'g'ri qatorlarni qo'shish buyrug'ida bajaramiz:

```
test=*# INSERT INTO groups(g_no, monitor)
SELECT 'A-101', s_id
FROM students
WHERE name = 'Анна';

INSERT 0 1
```

Taklifdagi “yulduzcha” kutilayotgan tranzaksiyani eslatadi.

Endi yangi terminal oynasini oching va boshqa `psql` jarayonini boshlang: bu birinchisi bilan parallel ravishda ishlaydigan seans bo'ladi. Chalkashmaslik uchun biz ikkinchi seansning buyruqlarini chizilgan holda ko‘rsatamiz.

Ikkinchi sessiya kiritilgan o‘zgarishlarni ko‘radimi?

```
postgres=# \c test
You are now connected to database "test" as user
"postgres".

test=# SELECT * FROM groups;

 g_no | monitor
------+---------
(0 rows)
```

Yo‘q, u buni ko'rmaydi, chunki tranzaksiya hali tugallanmagan.

Endi barcha talabalarni yaratilgan guruhga o‘tkazamiz:

```
test=*# UPDATE students SET g_no = 'A-101';

UPDATE 3
```

Va yana, ikkinchi seans hali ham tugallanmagan tranzaksiya boshida mavjud bo'lgan izchil ma'lumotlarni ko'radi:

```
test=# SELECT * FROM students;
 s_id |  name  | start_year | g_no
------+--------+------------+------
 1451 | Анна   |       2014 |
 1432 | Виктор |       2014 |
 1556 | Нина   |       2015 |
(3 rows)
```

Endi kiritilgan barcha o'zgarishlarni amalga oshirgan holda tranzaksiyani yakunlaymiz:

```
test=*# COMMIT;

COMMIT
```

Faqat shu daqiqada tranzaksiyaga kiritilgan barcha o'zgarishlar xuddi bir vaqtning o'zida paydo bo'lgandek, ikkinchi seansda mavjud bo'ladi:

```
test=# SELECT * FROM groups;
  g_no | monitor
-------+---------
 A-101 |    1451
(1 row)

test=# SELECT * FROM students;

 s_id |  name  | start_year | g_no
------+--------+------------+-------
 1451 | Анна   |       2014 | A-101
 1432 | Виктор |       2014 | A-101
 1556 | Нина   |       2015 | A-101
(3 rows)
```

MBBT bir nechta juda muhim kafolatlarni beradi.

Birinchidan, har qanday tranzaksiya yoki to'liq amalga oshiriladi (bizning misolimizda bo'lgani kabi) yoki u umuman bajarilmaydi. Agar buyruqlardan birini bajarishda xatolik yuzaga kelsa yoki biz o'zimiz `ROLLBACK` buyrug'i bilan tranzaksiyani to'xtatgan bo'lsak, ma'lumotlar bazasi `BEGIN` buyrug'idan oldingi holatda qoladi. Bu xususiyat **atomlik** deb ataladi.

Ikkinchidan, o'zgarishlarni amalga oshirishda barcha yaxlitlik cheklovlari bajarilishi kerak, aks holda tranzaksiya to'xtatiladi. Tranzaktsiyaning boshida ham, operatsiya oxirida ham ma'lumotlar, albatta, izchil holatda bo'ladi; Bu xususiyat **izchillik** deb ataladi.

Uchinchidan, biz misolda ko'rganimizdek, boshqa foydalanuvchilar tranzaksiya tomonidan hali amalga oshirilmagan ma'lumotlarni hech qachon ko'rmaydilar. Bu xususiyat **izolyatsiya** deb ataladi; Muvofiqligi tufayli MBBT ma'lumotlarning to'g'riligini yo'qotmasdan ko'plab seanslarga parallel ravishda xizmat ko'rsatishga qodir. PostgreSQL-ning o'ziga xos xususiyati uning izolyatsiyani juda samarali amalga oshirishidir: bir nechta seanslar bir vaqtning o'zida bir-birini bloklamasdan ma'lumotlarni o'qishi va o'zgartirishi mumkin. Bloklash faqat bir nechta turli jarayonlar bir vaqtning o'zida bir qatorni o'zgartirishga harakat qilganda paydo bo'ladi.

To'rtinchidan, **abadiylik** kafolatlanadi: yozib olingan ma'lumotlar hatto ishlamay qolgan taqdirda ham yo'qolmaydi (albatta, to'g'ri sozlamalar va muntazam zaxira nusxalari kabi amallar bilan).

Bu juda foydali xususiyatlar bo'lib, ularsiz relyatsion ma'lumotlar bazasi tizimini tasavvur qilib bo'lmaydi.

Tranzaktsiyalar haqida ko'proq ma'lumot olish uchun postgrespro.ru/doc/tutorial-transactionsga qarang (va undan ham ko'proq – postgrespro.ru/doc/mvcc).

Psql foydali buyruqlari

?	`psql` buyrug'i haqida yordam olish.
h	SQL Yordam: Mavjud buyruqlar ro'yxati yoki muayyan buyruq uchun sintaksis.
x	Jadval chiqishining holatini almashinuvi: oddiy holatdan (ustunlar va qatorlar) kengaytirilgan holatga (har bir ustun alohida satrda) va orqaga o'tkazish. Bir nechta "keng" qatorlarni ko'rish uchun qulay.
l	Ma'lumotlar bazalari ro'yxati.
du	Foydalanuvchilar ro'yxati.
dt	Jadvallar ro'yxati.
di	Indekslar ro'yxati.
dv	Ko'rinishlar ro'yxati.
df	Funksiyalar ro'yxati.
dn	Sxemalar ro'yxati.
dx	O'rnatilgan kengaytmalar ro'yxati.
dp	Imtiyozlar ro'yxati.
d nom	Muayyan ma'lumotlar bazasi obyekti haqida batafsil ma'lumot.
d+ nom	Muayyan obyekt haqida yanada batafsilroq ma'lumot.
timing on	Operatorlarning bajarilish vaqtlarini ko'rsatish.

Xulosa

Albatta, biz ma'lumotlar bazasi haqida bilishingiz kerak bo'lgan narsalarning ozgina qismini qamrab oldik, ammo PostgreSQL-dan foydalanishni boshlash unchalik qiyin emasligiga ishonchingiz komil degan umiddamiz. SQL tili har xil murakkablikdagi so'rovlarni shakllantirish imkonini beradi va PostgreSQL yuqori sifatli standart qo'llab-quvvatlash va samarali amalga oshirishni ta'minlaydi. Sinab ko'ring, tajriba!

Va yana bir muhim `psql` buyrug'i, seansni tugatish uchun:

```
test=# \q
```

V Demo ma‘lumotlar bazasi

Tavsif

Umumiy tavsif

Murakkab so‘rovlar bilan ishlashni o‘rganish uchun bizga ishonchli ma‘lumotlar bilan to‘ldirilgan uchta jadval emas, balki sakkizta jiddiyroq ma‘lumotlar bazasi kerak bo‘ladi.

Biz o‘z sohamiz sifatida havo transportini tanladik. Bizning ma‘lumotlar bazamizda ma‘lum vaqt davomida xayoliy aviakompaniya tomonidan amalga oshirilgan parvozlar haqida ma‘lumotlar mavjud. Samolyotda kim kamida bir marta uchgan bo‘lsa hamma narsa aniq bo‘lishi kerak, ammo har holda biz hamma narsani tushuntiramiz.

Bizning ma‘lumotlar bazamiz diagrammasi 69 betdagi rasmda ko‘rsatilgan. Biz keraksiz tafsilotlarga yo‘l qo‘ymaslik va shu bilan birga uning misolidan foydalangan holda qiziqarli va mazmunli so‘rovlarni ishlab chiqish uchun ma‘lumotlar sxemasini juda murakkab emas, lekin juda oddiy emas qilishga harakat qildik.

Asosiy ob‘ekt – har biriga alohida **chipta** (`tickets`) beriladigan bir yoki bir nechta yo‘lovchilar uchun **bron qilish** (`bookings`). Biz yo‘lovchini shaxs sifatida (u bizning aviakompaniyamiz bilan bir necha marta uchgan bo‘lishi mumkin) mustaqil mazmun sifatida ajrata olmaymiz, chunki u uchun

ishonchli yagona identifikator yo'q. Biz har bir yo'lovchi reysini unikal deb hisoblaymiz.

Har bir chipta bir yoki bir nechta **uchishlarga** mos keladi (`ticket_flights`). Chiptaga bir nechta reyslar – jo'nash va boradigan punktlar o'rtasida to'g'ridan-to'g'ri uchish mavjud bo'lmaganda (ya'ni, transfer bilan chipta olinadi) yoki ular bir-biriga to'g'ri kelganda (ya'ni, ikki tomonga, borish va qaytishga chipta olinganda) kiritilgan bo'ladi.

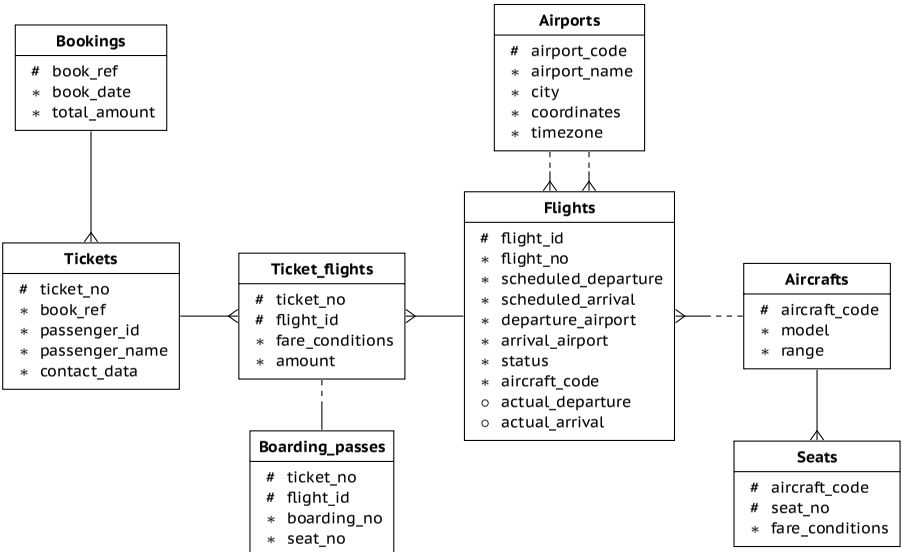
Bitta bronlashdagi barcha chiptalar bir xil reyslar to'plamiga ega deb taxmin qilinadi, garchi ma'lumotlar sxemasida bu borada qat'iy cheklov yo'q.

Har bir **reys** (`flights`) bir **aeroportdan** (`airports`) boshqasiga boradi. Bir xil raqamlarga ega reyslar bir xil jo'nash va kelish punktlariga ega, ammo turli sanalar.

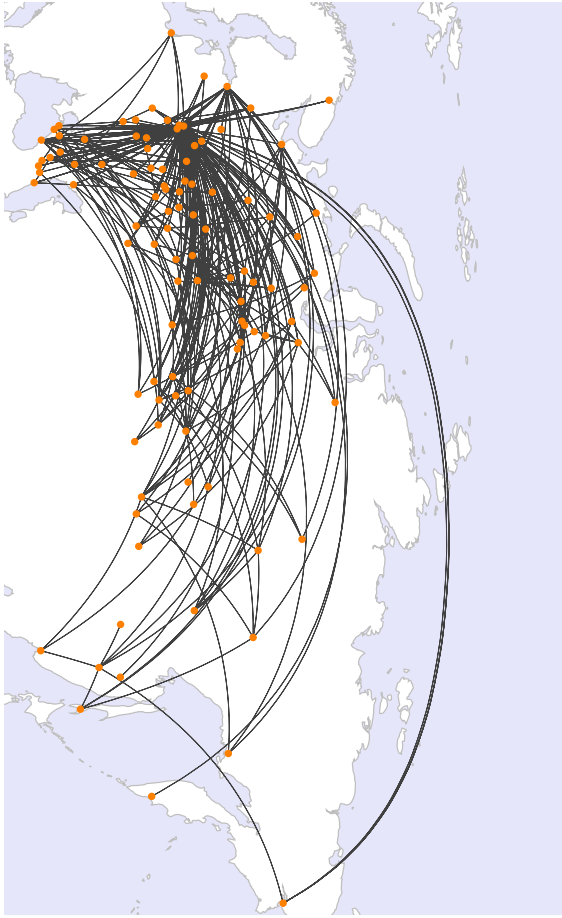
Parvozni ro'yxatdan o'tkazishda yo'lovchiga samolyotda o'tirgan joyini ko'rsatadigan **bortga chiqish talonlari** (`boarding_passes`) beriladi. Agar sizda chipta bo'lsa, siz faqat reysga ro'yxatdan o'tishingiz mumkin. Parvoz va o'rindiq kombinatsiyasi bir xil o'rindiq uchun turli xil bortga chiqish talonlari berilmasligini ta'minlash uchun unikal bo'lishi kerak.

Samolyotdagi **o'rindiqlar** (`seats`) soni va ularning xizmat ko'rsatish sinfi bo'yicha taqsimlanishi parvozni amalga oshiruvchi **samolyot** (`aircrafts`) modeliga bog'liq. Xuddi shu modeldagi samolyotlar bir xil kabina tartibiga ega deb, taxmin qilinadi. Ma'lumotlar sxemasi bortga chiqish talonida ko'rsatilgan o'rindiq mavjudligini tekshirishni nazarda tutmaydi.

Keyinchalik, har bir jadvalni, shuningdek, qo'shimcha ko'rinishlar va funktsiyalarni batafsil tavsiflaymiz. Har qanday


Bookings
book_ref
∗ book_date
∗ total_amount
Airports
airport_code
∗ airport_name
∗ city
∗ coordinates
∗ timezone
Tickets
ticket_no
∗ book_ref
∗ passenger_id
∗ passenger_name
∗ contact_data
Ticket_flights
ticket_no
flight_id
∗ fare_conditions
∗ amount
Flights
flight_id
∗ flight_no
∗ scheduled_departure
∗ scheduled_arrival
∗ departure_airport
∗ arrival_airport
∗ status
∗ aircraft_code
○ actual_departure
○ actual_arrival
Aircrafts
aircraft_code
∗ model
∗ range
Boarding_passes
ticket_no
flight_id
∗ boarding_no
∗ seat_no
Seats
aircraft_code
seat_no
∗ fare_conditions

70
v



jadvalning aniq ta'rifi, shu jumladan ma'lumotlar turlari va ustunlarning to'liq tavsiflari har doim `\d+` buyruq yordamida olinishi mumkin.

Bronlash

Aviakompaniyamiz xizmatlaridan foydalanish niyatida yo'lovchi oldindan kerakli chiptalarni band qilib qo'yadi (`book_date`, parvozdan maksimal bir oy oldin). Bronlash `book_ref` raqami bilan identifikatsiyalanadi (harflar va raqamlarning olti xonali birikmasi).

`total_amount` maydonida barcha yo'lovchilar uchun bronga kiritilgan reyslarning umumiy qiymati saqlanadi.

Chipta

Chiptada 13 ta raqamdan iborat `ticket_no` unikal raqami mavjud.

Chipta `passenger_id` yo'lovchini shaxsligini identifikatsiya qiluvchi hujjat raqami, shuningdek uning familiyasi va ismi `passenger_name` va bog'lanish ma'lumotlari `contact_data`-ni o'z ichiga oladi.

Yo'lovchining guvohnomasi ham, uning ismi ham doimiy emasligini unutmang (pasportingizni o'zgartirishingiz mumkin, familiyangizni o'zgartirishingiz mumkin). Shu sababli, bitta yo'lovchi uchun barcha chiptalarni aniqlab bo'lmaydi. Oddiylik uchun barcha yo'lovchilar unikal deb taxmin qilishimiz mumkin.

Parvoz

Parvoz chiptani parvoz bilan bog'laydi va ularning ikkita raqami bilan belgilanadi.

Har bir parvoz uchun uning narxi `amount` va tariflar klassi `fare_conditions` ko'rsatilgan.

Reys

Unikal identifikator tabiiy (agar u real ob'ektlarning xususiyatlarini o'z ichiga olgan bo'lsa) yoki o'gay (agar tizim tomonidan yaratilgan bo'lsa, odatda raqamlarning ortib borayotgan ketma-ketligidan) bo'lishi mumkin.

Parvoz jadvalining tabiiy kaliti `flight_no` reys raqami va `scheduled_departure` sanasidan iborat. Ushbu jadvaldagi tashqi kalitlarni yanada ixcham qilish uchun birlamchi kalit sifatida `flight_id` o'gay kalit ishlatiladi.

Parvoz har doim ikkita nuqtani bog'laydi: `departure_airport` va `arrival_airport`.

"Transferlar bilan parvoz" degan narsa yo'q: agar aeroportlar o'rtasida to'g'ridan-to'g'ri reys bo'lmasa, barcha kerakli reyslar shunchaki chiptaga kiritilgan.

Har bir reysda jo'nash `scheduled_departure` va kelishlarning (`scheduled_arrival`) rejalashtirilgan sanasi va vaqti mavjud. Haqiqiy jo'nash vaqti – `actual_departure` va haqiqiy kelish vaqti – `actual_arrival` farq qilishi mumkin; ko'pincha ko'p emas, lekin ba'zida parvozning kechikishi bir necha soat bo'lishi mumkin.

Parvoz holati `status` quyidagi qiymatlarni olishi mumkin:

- Scheduled
  Reysni bron qilish mumkin. Qiymat rejalashtirilgan jo'nash sanasidan bir oy oldin, ma'lumotlar bazasida parvoz yozuvi paydo bo'lishi bilan bir vaqtda o'rnatiladi.
- On Time
  Reys ro'yxatdan o'tish uchun tayyor (belgilangan jo'nash sanasidan bir kun oldin) va kechiktirilmaydi.
- Delayed
  Reys ro'yhatdan o'tish uchun tayyor (belgilangan jo'nash sanasidan bir kun oldin), lekin kechiktirilgan.
- Departed
  Samolyot allaqachon havoga ko'tarilgan va havoda.
- Arrived
  Samolyot belgilangan manzilga yetib keldi.
- Cancelled
  Reys bekor qilingan.

Aeroport

Har bir aeroport uch harfli `airport_code` bilan belgilanadi va `airport_name` nomga ega.

Shahar nomi `city` aeroport atributi sifatida ko'rsatilgan; uning uchun alohida ob'ekt yo'q. Ular bitta shaharning aeroportlarini aniqlashda kerak. Koordinatalar (uzunlik va kenglik) `coordinates` va `timezone` ham ko'rsatilgan.

Bortga chiqish taloni

Belgilangan jo'nash kunidan bir kun oldin, mumkin bo'lgan reysga ro'yxatdan o'tishda yo'lovchiga bortga chiqish taloni

beriladi. U xuddi reys bilan bir xil – chipta va parvoz raqami bo'yicha identifikatsiyalanadi.

Bortga chiqish talonlariga yo'lovchilar reysga ro'yxatdan o'tish tartibi bo'yicha ketma-ket `boarding_no` raqamlari beriladi (bu raqam faqat ma'lum bir reys doirasida yagona bo'ladi). Bortga chiqish talonida o'rindiq raqami `seat_no` ko'rsatilgan.

Samolyot

Har bir samolyot modeli uch xonali `aircraft_code` bilan aniqlanadi. Shu bilan birga modelning nomi – `model` va kilometrlarda maksimal parvoz masofasi – `range` ham ko'rsatilgan bo'ladi.

Joy

O'rindiqlar to'plami har bir modelning ichki tuzlishini belgilaydi. Har bir o'rindiq `seat_no` raqami bilan belgilanadi va unga `fare_conditions` xizmat ko'rsatish sinfi tayinlangan – Economy, Comfort yoki Business.

Parvozlar uchun ko'rinish

`flights_v` ko'rinishi `flights` jadvali ustida yaratilgan. Ko'rinishlar xuddi jadvallar kabi so'rovlarda ishlatilishi mumkin, lekin ma'lumotlarni saqlash o'rniga ular qandaydir so'rovlarni bajaradilar. Ko'rinish ta'rifi va u tomonidan saqlangan so'rovni quyidagi psql buyrug'i bilan ko'rish mumkin:

```
postgres=# \d+ flights_v
```

Ushbu ko‘rinish quyidagi ma‘lumotlarni qo‘shadi:

- `departure_airport`, `departure_airport_name`, `departure_city` kabi aeroport jo‘nab ketish haqidagi ma‘lumotlarini dekodlash;
- `arrival_airport`, `arrival_airport_name`, `arrival_city` kabi aeroport qo‘nish haqidagi ma‘lumotlarini dekodlash;
- uchish mahalliy vaqti `scheduled_departure_local`, `actual_departure_local`;
- qo‘nish mahalliy vaqti `scheduled_arrival_local`, `actual_arrival_local`;
- uchish davomiyligi `scheduled_duration`, `actual_duration`.

Marshrutlar uchun ko‘rinishlar

Parvoz jadvalida ortiqcha ma‘lumotlar mavjud: undan aniq parvoz sanalariga bog‘liq bo‘lmagan marshrut (parvoz raqami, jo‘nash va boradigan aeroportlar, samolyot modeli) haqida ma‘lumot olish mumkin bo‘ladi.

Aynan shu ma‘lumotlar `routes` ko‘rinishini tashkil qiladi. Bundan tashqari, ushbu ko‘rinish reyslar amalga oshiriladigan haftaning kunlarini `days_of_week` va parvoz davomiyligining rejalashtirilgan qiymati `duration` ko‘rsatadi.

Now funksiyasi

Demo ma‘lumotlar bazasi ma‘lumotlarning vaqt “bo‘lagini” o‘z ichiga oladi – go‘yo bir vaqtda haqiqiy tizimning zaxira nusxasi yaratilgan. Misol uchun, agar ma‘lum bir reysda

Departed statusi mavjud bo‘lsa, bu havo kemasi zaxirani nusxalash vaqtida havoda bo‘lganligini anglatadi.

“Tilim” joylashuvi `bookings.now` funksiyasida saqlanadi. U, odatda `now` funksiyasi qo‘llaniladigan so‘rovlarda ishlatilishi mumkin.

Bundan tashqari, ushbu funktsiyaning qiymati demo ma‘lumotlar bazasi talqinini aniqlaydi. Kitobning ushbu sonini tayyorlash vaqtidagi joriy talqini 15.08.2017dan boshlangan.

O‘rnatish

Saytdan o‘rnatish

Ma‘lumotlar bazasi uchta talqinda mavjud bo‘lib, ular faqat hajmi bo‘yicha farqlanadi:

- edu.postgrespro.ru/demo-small.zip – katta bo‘lmagan, parvozlar to‘g‘risidagi 1 oylik ma‘lumotlar (fayl – 21 MB, ma‘lumotlar bazasi hajmi 280 MB)),
- edu.postgrespro.ru/demo-medium.zip – o‘rtacha, 3 oylik parvozlar haqidagi ma‘lumotlar (fayl – 62 MB, ma‘lumotlar bazasi hajmi 702 MB),
- edu.postgrespro.ru/demo-big.zip – katta, 1 yillik parvozlar ma‘lumotlari (fayl – 232 MB, ma‘lumotlar bazasi hajmi 2638 MB).

Kichkina ma‘lumotlar bazasida so‘rovlar yozishni mashq qilishingiz mumkin, ammo optimallashtirish masalalariga sho‘ng‘ish uchun katta ma‘lumotlar yaxshiroq – keyin siz

so'rovlar katta hajmdagi ma'lumotlarda qanday harakat qilishini darhol ko'rasiz.

Fayllar `pg_dump` yordamchi dasturi tomonidan yaratilgan `demo` ma'lumotlar bazasining mantiqiy zaxira nusxasini o'z ichiga oladi. Esda tutingki, agar sizda allaqachon `demo` nomli ma'lumotlar bazasi mavjud bo'lsa, u zaxiradan qayta tiklanganda o'chiriladi va qayta yaratiladi. `demo` ma'lumotlar bazasining egasi qayta tiklashni amalga oshirgan MBBT foydalanuvchisi bo'ladi.

Demo ma'lumotlar bazasini Linux operatsion tizimiga o'rnatish uchun `postgres` foydalanuvchisiga o'tgandan so'ng fayllardan birini yuklab oling. Masalan, kichik ma'lumotlar bazasi quyidagicha yuklab olinadi:

```
$ sudo su - postgres
$ wget https://edu.postgrespro.ru/demo-small.zip
```

So'ngra quyidagini bajaring:

```
$ zcat demo-small.zip | psql
```

Windows operatsion tizimida istalgan veb-brauzerdan foydalanib, saytdan edu.postgrespro.ru/demo-small.zip faylini yuklab oling, so'ngra arxivni ochish uchun ustiga ikki marta bosing va keyin `demo-small-20170815.sql` faylidan `C:\Program Files\PostgreSQL\14` katalogiga nusxa oling.

pgAdmin dasturi (117-betda muhokama qilingan) bunday zaxira nusxasidan ma'lumotlar bazasini tiklashga ruxsat bermaydi. Shunday qilib, psql ("SQL Shell (psql)" yorlig'i)ni ishga tushiring va buyruqni bajaring:

```
postgres# \i demo-small-20170815.sql
```

Agar fayl topilmasa, yorliq xususiyatini tekshiring “Start in” (“Ishchi papka”) – fayl ushbu katalogda joylashgan bo'lishi kerak.

So'rovlar misollari

Sxema haqida

O'rnatish tugallandi. Psqlni ishga tushiring va demo ma'lumotlar bazasiga ulaning:

```
postgres=# \c demo

You are now connected to database "demo" as user
"postgres".
```

Bizni qiziqtirgan barcha ob'ektlar `bookings` sxemasida. Ma'lumotlar bazasiga ulanishda ushbu sxema avtomatik ravishda ishlatiladi, shuning uchun uni aniq ko'rsatishga hojat yo'q:

```
demo=# SELECT * FROM aircrafts;

 aircraft_code |        model         | range
---------------+----------------------+-------
 773           | Боинг 777-300        | 11100
 763           | Боинг 767-300        |  7900
 SU9           | Сухой Суперджет-100 |  3000
 320           | Аэробус A320-200     |  5700
 321           | Аэробус A321-200     |  5600
 319           | Аэробус A319-100     |  6700
 733           | Боинг 737-300        |  4200
 CN1           | Сессна 208 Караван   |  1200
 CR2           | Бомбардье CRJ-200    |  2700
(9 rows)
```

Ammo bookings.now funktsiyasini chaqirganda, bu kerak, chunki standart now funksiyasi ham mavjud:

```
demo=# SELECT bookings.now();

          now
------------------------
 2017-08-15 18:00:00+03
(1 row)
```

Siz allaqachon payqaganingizdek, samolyotlar, aeroportlar va shaharlarning nomlari rus tilida ko'rsatilgan:

```
demo=# SELECT airport_code, city
FROM airports LIMIT 5;

 airport_code |           city
--------------+-------------------------
 YKS          | Якутск
 MJZ          | Мирный
 KHV          | Хабаровск
 PKC          | Петропавловск-Камчатский
 UUS          | Южно-Сахалинск
(5 rows)
```

Agar tilni aniqlaydigan bookings.lang parametri enga o'rnatilgan bo'lsa, u ingliz tilida bo'ladi:

```
demo=# ALTER DATABASE demo SET bookings.lang = en;

ALTER DATABASE
```

Biz ma'lumotlar bazasi darajasida tilni o'zgartirdik; Endi qayta ulanishingiz kerak.

```
demo=# \c

You are now connected to database "demo" as user
"postgres".
```

```
demo=# SELECT airport_code, city
FROM airports
LIMIT 5;

 airport_code |       city
--------------+-------------------
 YKS          | Yakutsk
 MJZ          | Mirnyj
 KHV          | Khabarovsk
 PKC          | Petropavlovsk
 UUS          | Yuzhno-sakhalinsk
(5 rows)
```

Tilni o'zgartirish mexanizmini o'rganish uchun psql `\d+` buyrug'i bilan `aircrafts` yoki `airports`-larning ta'rifiga qarang.

Sxemani boshqarish haqida ko'proq bilib oling: postgrespro.ru/doc/ddl-schemas.

Server parametrlarini sozlash haqida — postgrespro.ru/doc/config-setting.

Oddiy so'rovlar

Keling, ushbu sxema uchun bir nechta vazifalarni ko'rib chiqaylik – eng oddiydan ancha murakkabgacha. Aksariyat muammolarning yechimlari bor, lekin avval javobga qaramasdan so'rov yozishga harakat qilgan ma'qul, chunki aks holda siz SQL tilini o'rgana olmaysiz.

Topshiriq. Kecha bir kun oldin Moskva (SVO) – Novosibirsk (OVB) reysida 1A o'rindig'ida kim uchdi va u chiptani qachon bron qilgan?

Yechim. "Kechagi kun" joriy sanadan emas, balki `booking.now`dan hisoblanadi.

```
SELECT t.passenger_name,
       b.book_date
FROM   bookings b
       JOIN tickets t
         ON t.book_ref = b.book_ref
       JOIN boarding_passes bp
         ON bp.ticket_no = t.ticket_no
       JOIN flights f
         ON f.flight_id = bp.flight_id
WHERE  f.departure_airport = 'SVO'
AND    f.arrival_airport = 'OVB'
AND    f.scheduled_departure::date =
       bookings.now()::date - INTERVAL '2 day'
AND    bp.seat_no = '1A';
```

Topshiriq. Kecha PG0404 reysida nechta o'rindiq bo'sh qoldi?

Yechim. Muammoni bir necha usul bilan hal qilish mumkin. Bitta yo'li, NOT EXISTS konstruksiyasidan foydalanib, biz bortga chiqish talonlarisiz bo'lgan o'rindiqlarni topamiz:

```
SELECT count(*)
FROM   flights f
       JOIN seats s
         ON s.aircraft_code = f.aircraft_code
WHERE  f.flight_no = 'PG0404'
AND    f.scheduled_departure::date =
       bookings.now()::date - INTERVAL '1 day'
AND    NOT EXISTS (
         SELECT NULL
         FROM   boarding_passes bp
         WHERE  bp.flight_id = f.flight_id
         AND    bp.seat_no = s.seat_no
       );
```

Boshqa variantda biz to'plamlarni ayirishga murojaat qilamiz. Turli xil yechimlar bir xil natija beradi, lekin ba'zida ishlashda farqlanadi, shuning uchun agar bu muhim bo'lsa, buni hisobga olishga arziydi.

```
SELECT count(*)
FROM   (
  SELECT s.seat_no
  FROM   seats s
  WHERE  s.aircraft_code = (
    SELECT aircraft_code
    FROM   flights
    WHERE  flight_no = 'PG0404'
    AND    scheduled_departure::date =
           bookings.now()::date - INTERVAL '1 day'
  )
  EXCEPT
  SELECT bp.seat_no
  FROM   boarding_passes bp
  WHERE  bp.flight_id = (
    SELECT flight_id
    FROM   flights
    WHERE  flight_no = 'PG0404'
    AND    scheduled_departure::date =
           bookings.now()::date - INTERVAL '1 day'
  )
) t;
```

Topshiriq. Qaysi reyslar eng uzoq kechikishlarga duch keldi? Eng uzoq muddatga kechiktirilgan o'nta reysni sanab o'ting.

Yechim. Biz faqat bajarilgan parvozlarni hisobga olamiz.

```
SELECT   f.flight_no,
         f.scheduled_departure,
         f.actual_departure,
         f.actual_departure - f.scheduled_departure
         AS delay
FROM     flights f
WHERE    f.actual_departure IS NOT NULL
ORDER BY f.actual_departure - f.scheduled_departure
         DESC
LIMIT 10;
```

Xuddi shu shart `status` ustuni asosida yozilishi mumkin, barcha mos keladigan holatlar ro'yxati. Yoki `DESC NULLS LAST`

tartiblash tartibini belgilash orqali WHERE shartisiz bajarishingiz mumkin, shunda aniqlanmagan qiymatlar tanlov boshida emas, balki oxirida tugaydi.

Agregat funksiyalar

Topshiriq. Moskvadan Sankt-Peterburgga har bir mumkin bo'lgan parvoz uchun minimal va maksimal parvoz davomiyligi qancha va parvoz necha marta bir soatdan ortiq kechiktirildi?

Yechim. Bu yerda kerakli jadvallarning ulanishlarini yozmaslik uchun tayyor `flights_v` ko'rnishidan foydalanish qulay. So'rovda biz faqat allaqachon tugallangan reyslarni hisobga olamiz.

```
SELECT   f.flight_no,
         f.scheduled_duration,
         min(f.actual_duration),
         max(f.actual_duration),
         sum(CASE WHEN f.actual_departure >
                       f.scheduled_departure +
                       INTERVAL '1 hour'
                  THEN 1 ELSE 0
             END) delays
FROM     flights_v f
WHERE    f.departure_city = 'Москва'
AND      f.arrival_city = 'Санкт-Петербург'
AND      f.status = 'Arrived'
GROUP BY f.flight_no,
         f.scheduled_duration;
```

Topshiriq. Barcha reyslarga birinchi bo'lib ro'yxatdan o'tgan eng intizomli yo'lovchilarni toping. Faqat kamida ikkita reysni amalga oshirgan yo'lovchilarni hisobga oling.

Yechim. Biz bortga chiqish talonlari raqamlari ro'yxatdan o'tish tartibida berilishidan foydalanamiz.

```
SELECT   t.passenger_name,
         t.ticket_no
FROM     tickets t
         JOIN boarding_passes bp
           ON bp.ticket_no = t.ticket_no
GROUP BY t.passenger_name,
         t.ticket_no
HAVING   max(bp.boarding_no) = 1
AND      count(*) > 1;
```

Topshiriq. Har bir bron qilish uchun qancha yo'lovchi mos keladi?

Yechim. Keling, avval har bir bronlashda yo'lovchilar sonini, so'ngra yo'lovchilar sonining har bir varianti bilan bandlar sonini hisoblaymiz.

```
SELECT   tt.cnt,
         count(*)
FROM     (
           SELECT   t.book_ref,
                    count(*) cnt
           FROM     tickets t
           GROUP BY t.book_ref
         ) tt
GROUP BY tt.cnt
ORDER BY tt.cnt;
```

Oyna funksiyalar

Topshiriq. Har bir chipta uchun unga kiritilgan reyslarni va keyingi reysga o'tish uchun qolgan zaxira vaqtni ham ko'rsating. Saralashingizni yetti kun oldin bron qilingan chiptalar bilan cheklang.

Yechim. Xuddi shu ma'lumotlarga ikki marta kirishni oldini olish uchun biz oyna funksiyalaridan foydalanamiz.

```
SELECT tf.ticket_no,
       f.departure_airport,
       f.arrival_airport,
       f.scheduled_arrival,
       lead(f.scheduled_departure) OVER w
       AS next_departure,
       lead(f.scheduled_departure) OVER w -
         f.scheduled_arrival
       AS gap
FROM   bookings b
       JOIN tickets t
         ON t.book_ref = b.book_ref
       JOIN ticket_flights tf
         ON tf.ticket_no = t.ticket_no
       JOIN flights f
         ON tf.flight_id = f.flight_id
WHERE  b.book_date =
       bookings.now()::date - INTERVAL '7 day'
WINDOW w AS (
         PARTITION BY tf.ticket_no
         ORDER BY f.scheduled_departure
       );
```

Ko'rib turganingizdek, ba'zi hollarda transfer zahira vaqti bir necha kungacha bo'lishi mumkin, chunki ikki tomonlama chiptalar bir tomonlama chiptalar bilan bir xil asosda hisoblanadi, belgilangan joyga sarflangan vaqt esa transfer vaqti bilan bir xilda hisobga olinadi. “Massivlar” bo‘limidagi muammolardan birini hal qilishdan foydalanib, siz ushbu ravshanlikni so‘rovda hisobga olishingiz mumkin.

Topshiriq. Ism va familiyaning qanday kombinatsiyasi eng keng tarqalgan va ular barcha yo'lovchilarning qaysi qismini tashkil qiladi?

Yechim. Yo'lovchilarning umumiy sonini hisoblash uchun oyna funksiyasidan foydalaniladi.

```
SELECT   passenger_name,
         round( 100.0 * cnt / sum(cnt) OVER (), 2)
         AS percent
FROM     (
           SELECT   passenger_name,
                    count(*) cnt
           FROM     tickets
           GROUP BY passenger_name
         ) t
ORDER BY percent DESC;
```

Topshiriq. Oldingi masalani ismlar uchun alohida va familiyalar uchun alohida hal qiling.

Yechim. Bu yerda nomlarni sanash varianti mavjud. Familiyalar uchun yechim faqat p qismsoʻrovida farqlanadi.

```
WITH p AS (
  SELECT left(passenger_name,
              position(' ' IN passenger_name))
         AS passenger_name
  FROM   tickets
)
SELECT   passenger_name,
         round( 100.0 * cnt / sum(cnt) OVER (), 2)
         AS percent
FROM     (
           SELECT   passenger_name,
                    count(*) cnt
           FROM     p
           GROUP BY passenger_name
         ) t
ORDER BY percent DESC;
```

Xulosa shuki: agar siz ular bilan alohida ishlamoqchi boʻlsangiz, bitta matn maydonida bir nechta qiymatlarni birlashtirmang. Ilmiy jihatdan bu “birinchi normal shakl” deb ataladi.

Topshiriq. Chiptada uning bir tomonlama yoki ikki tomonlama ekanligi aniq ko'rsatilmagan. Ammo buni birinchi jo'nash nuqtasini oxirgi manzil bilan solishtirish orqali aniqlash mumkin. Har bir chipta uchun jo'nash va boradigan aeroportlarni ko'rsating, transferlar bundan mustasno va uning bir tomonlama yoki ikki tomonlama ekanligini ko'rsating.

Yechim. Ehtimol, eng oson yo'li bu – `array_agg` agregat funksiyasidan foydalangan holda marshrut bo'ylab aeroportlar ro'yxatini massivga yig'ish va u bilan ishlash bo'lishi mumkin.

Ikki tomonga chiptalar uchun belgilangan aeroportlarni massivining o'rta elementini tanlaymiz, bunda borish va qaytishlardagi transferlar soni bir xilligini hisobga olamiz.

```
WITH t AS (
  SELECT ticket_no,
         a,
         a[1]                  departure,
         a[cardinality(a)]     last_arrival,
         a[cardinality(a)/2+1] middle
  FROM (
    SELECT   t.ticket_no,
             array_agg( f.departure_airport
               ORDER BY f.scheduled_departure) ||
            (array_agg( f.arrival_airport
               ORDER BY f.scheduled_departure DESC)
            )[1] AS a
    FROM     tickets t
             JOIN ticket_flights tf
               ON tf.ticket_no = t.ticket_no
             JOIN flights f
               ON f.flight_id = tf.flight_id
    GROUP BY t.ticket_no
  ) t
)
```

```
SELECT t.ticket_no,
       t.a,
       t.departure,
       CASE
         WHEN t.departure = t.last_arrival
           THEN t.middle
         ELSE t.last_arrival
       END arrival,
       (t.departure = t.last_arrival) return_ticket
FROM   t;
```

Ushbu parametrda chiptalar jadvali faqat bir marta ko'riladi. Aeroportlar massivi faqat ko'rish uchun ko'rsatiladi; Agar katta hajmdagi ma'lumotlar mavjud bo'lsa, uni so'rovdan olib tashlash mantiqan to'g'ri keladi, chunki qo'shimcha ma'lumotlar ishlash tezligiga yaxshi ta'sir ko'rsatmasligi mumkin.

Topshiriq. "U yerga" yo'li "ortga" yo'li bilan bir xil bo'lmagan ikki tomonlama chiptalarni toping.

Topshiriq. Haftaning turli kunlarida bir yo'nalishda va boshqa yo'nalishda parvoz qiladigan aeroportlar juftligini toping.

Yechim. Haftaning bir qator kunlarini yaratish vazifasining bir qismi allaqachon `routes` ko'rinishida hal qilingan. Faqat && operatori yordamida massivlarning kesishishini topish va uning bo'shligiga ishonch hosil qilish qoladi:

```
SELECT r1.departure_airport,
       r1.arrival_airport,
       r1.days_of_week dow,
       r2.days_of_week dow_back
FROM   routes r1
       JOIN routes r2
         ON r1.arrival_airport = r2.departure_airport
        AND r1.departure_airport = r2.arrival_airport
WHERE  NOT (r1.days_of_week && r2.days_of_week);
```

Topshiriq. Ust-Kutdan (UKX) Neryungriga (CNN) minimal transferlar bilan qanday borish mumkin va havoda qancha vaqt sarflashingiz kerak?

Yechim. Bu yerda siz aslida grafikdagi eng qisqa yo'lni topishingiz kerak, bu rekursiv so'rov bilan amalga oshiriladi.

```
WITH RECURSIVE p(
  last_arrival,
  destination,
  hops,
  flights,
  flight_time,
  found
) AS (
  SELECT a_from.airport_code,
         a_to.airport_code,
         array[a_from.airport_code],
         array[]::char(6)[],
         interval '0',
         a_from.airport_code = a_to.airport_code
  FROM   airports a_from,
         airports a_to
  WHERE  a_from.airport_code = 'UKX'
  AND    a_to.airport_code = 'CNN'
  UNION ALL
  SELECT r.arrival_airport,
         p.destination,
         (p.hops || r.arrival_airport)::char(3)[],
         (p.flights || r.flight_no)::char(6)[],
         p.flight_time + r.duration,
         bool_or(r.arrival_airport = p.destination)
           OVER ()
  FROM   p
         JOIN routes r
           ON r.departure_airport = p.last_arrival
  WHERE  NOT r.arrival_airport = ANY(p.hops)
  AND    NOT p.found
)
```

```
SELECT hops,
       flights,
       flight_time
FROM   p
WHERE  p.last_arrival = p.destination;
```

Ushbu so'rov habr.com/company/postgrespro/blog/318398 maqolasida batafsil, bosqichma-bosqich muhokama qilinadi, shuning uchun biz bu yerda faqat qisqacha sharhlar beramiz.

So'rovni bajarish paytida qurilgan qo'nishlar `hops` massivi bo'yicha tekshirishlar yordamida aylanishning oldi olinadi.

E'tibor bering, qidiruv "eniga", ya'ni birinchi topilgan yo'l transferlar soni bo'yicha eng qisqa bo'ladi. Boshqa yo'llardan o'tmaslik uchun (ular ko'p bo'lishi mumkin va ular allaqachon topilganidan uzunroq) "marshrut topildi" (`found`) atributidan foydalaniladi. U `bool_or` oyna funksiyasi yordamida hisoblanadi.

Ushbu so'rovning bajarilish tezligini bayroqsiz oddiyroq variant bilan solishtirish juda foydali.

Rekursiv so'rovlar haqida batafsil ma'lumotni hujjatlarda topishingiz mumkin: postgrespro.ru/doc/queries-with.

Topshiriq. Bir aeroportdan boshqasiga o'tish uchun maksimal qancha transferlar talab qilinishi mumkin?

Yechim. Yechim uchun asos sifatida oldingi so'rovni olishingiz mumkin. Ammo endi dastlabki iteratsiya faqat bir juft aeroportni emas, balki barcha mumkin bo'lgan juftlarni o'z ichiga olishi kerak: biz har bir aeroportni har biriga ulaymiz. Barcha bunday juftliklar uchun biz eng qisqa yo'lni qidiramiz va keyin eng kattasini tanlaymiz.

Albatta, bu faqat marshrut grafigi ulangan bo'lsa amalga oshirilishi mumkin, ammo bizning demo ma'lumotlar bazasida bu shart bajariladi.

Ushbu so'rovda "marshrut topildi" atributi ham qo'llaniladi, ammo bu yerda u har bir aeroport juftligi uchun alohida hisoblanishi kerak.

```
WITH RECURSIVE p(
  departure,
  last_arrival,
  destination,
  hops,
  found
) AS (
  SELECT a_from.airport_code,
         a_from.airport_code,
         a_to.airport_code,
         array[a_from.airport_code],
         a_from.airport_code = a_to.airport_code
  FROM   airports a_from,
         airports a_to
  UNION ALL
  SELECT p.departure,
         r.arrival_airport,
         p.destination,
         (p.hops || r.arrival_airport)::char(3)[],
         bool_or(r.arrival_airport = p.destination)
           OVER (PARTITION BY p.departure,
                              p.destination)
  FROM   p
         JOIN routes r
           ON r.departure_airport = p.last_arrival
  WHERE  NOT r.arrival_airport = ANY(p.hops)
  AND    NOT p.found
)
SELECT max(cardinality(hops)-1)
FROM   p
WHERE  p.last_arrival = p.destination;
```

Topshiriq. Ust-Kutdan (UKX) Neryungriga (CNN) eng qisqa marshrutni aniq parvoz vaqti bo'yicha toping (qo'nish vaqtini hisobga olmaganda).

Maslahat: bu marshrut transferlar soni bo'yicha optimal bo'lmasligi mumkin.

Yechim. Uzluksiz siklning oldini olish uchun PostgreSQL 14da kiritilgan CYCLE konstruktsiyasi qo'llaniladi.

```
WITH RECURSIVE p(
  last_arrival,
  destination,
  flights,
  flight_time,
  min_time
) AS (
  SELECT a_from.airport_code,
         a_to.airport_code,
         array[]::char(6)[],
         interval '0',
         NULL::interval
  FROM   airports a_from,
         airports a_to
  WHERE  a_from.airport_code = 'UKX'
  AND    a_to.airport_code = 'CNN'
  UNION ALL
  SELECT r.arrival_airport,
         p.destination,
         (p.flights || r.flight_no)::char(6)[],
         p.flight_time + r.duration,
         least(
           p.min_time,
           min(p.flight_time + r.duration)
           FILTER (
             WHERE r.arrival_airport = p.destination
           ) OVER ()
         )
  FROM   p
         JOIN routes r
           ON r.departure_airport = p.last_arrival
```

```
  WHERE  p.flight_time + r.duration
         < coalesce(
             p.min_time,
             INTERVAL '1 year'
           )
)
CYCLE last_arrival SET is_cycle USING hops
SELECT hops,
       flights,
       flight_time
FROM   (
         SELECT hops,
                flights,
                flight_time,
                min(min_time) OVER () min_time
         FROM   p
         WHERE  p.last_arrival = p.destination
       ) t
WHERE  flight_time = min_time;
```

Funksiyalar va kengaytmalar

Topshiriq. Kaliningrad (KGD) va Petropavlovsk-Kamchatskiy (PKC) o'rtasidagi masofani toping.

Yechim. `airports` jadvali aeroportlarning koordinatalarini o'z ichiga oladi. Juda uzoq nuqtalar orasidagi masofani aniq hisoblash uchun siz yerning murakkab shaklini hisobga olishingiz kerak. Yaxshisi bu ishni yaxshi hal qiluvchi PostGIS kengaytmasidan foydalanilgani qulay, u yer yuzasini geoid bilan yaqinlashtirishi mumkin.

Ammo bizning maqsadlarimiz uchun oddiy sharsimon model ham ishlaydi. Keling, `earthdistance` kengaytmasidan foydalanamiz va natijani mildan kilometrga aylantiramiz.

```
CREATE EXTENSION IF NOT EXISTS cube;
```

```
CREATE EXTENSION IF NOT EXISTS earthdistance;

SELECT round(
         (a_from.coordinates <@> a_to.coordinates) *
         1.609344
       )
FROM   airports a_from,
       airports a_to
WHERE  a_from.airport_code = 'KGD'
AND    a_to.airport_code = 'PKC';
```

Topshiriq. Aeroportlar orasidagi parvozlar grafigini chizing.

VI PostgreSQL ilovalar uchun

Alohida foydalanuvchi

Oldingi bobda biz ma'lumotlar bazasi serveriga MBBT o'rnatilgandan so'ng darhol mavjud bo'lgan `postgres` foydalanuvchisi sifatida ulandik. Ammo `postgres` superfoydalanuvchi huquqlariga ega va siz ushbu huquqlarga ega bo'lgan dastur orqali ma'lumotlar bazasiga ulanmasligingiz kerak. Yangi foydalanuvchi yaratib, uni alohida ma'lumotlar bazasi egasi qilib qo'ygan ma'qul – keyin uning imkoniyatlari ushbu ma'lumotlar bazasi bilan cheklanadi.

```
postgres=# CREATE USER app PASSWORD 'p@ssw0rd';

CREATE ROLE

postgres=# CREATE DATABASE appdb OWNER app;

CREATE DATABASE
```

Foydalanuvchilar va imkoniyatlarni boshqarish haqida qo'shimcha ma'lumot olish uchun ushbu murojatga qarang: postgrespro.ru/doc/user-manag va postgrespro.ru/doc/ddl-priv.

Yangi ma'lumotlar bazasiga ulanish va yaratilgan foydalanuvchi nomidan u bilan ishlash uchun quyidagilarni bajaring:

```
postgres=# \c appdb app localhost 5432

Password for user app: ***
You are now connected to database "appdb" as user
"app" on host "127.0.0.1" at port "5432".

appdb=>
```

Buyruqda ketma-ket ma'lumotlar bazasi nomini (appdb), foydalanuvchi nomini (app), xostni (localhost yoki 127.0.0.1) va port raqamini (5432) ko'rsatiladi.

E'tibor bering, so'rovda faqat ma'lumotlar bazasi nomi o'zgargan emas, endi u "xesh" o'rniga "katta" belgisini ko'rsatadi. Xesh belgilari Unix operatsion tizimidagi `root` foydalanuvchiga o'xshash superfoydalanuvchi rolini bildiradi.

`app` foydalanuvchisi o'z ma'lumotlar bazasi bilan cheklovlarsiz ishlaydi. Masalan, unda jadval yaratishingiz mumkin:

```
appdb=> CREATE TABLE greeting(s text);

CREATE TABLE

appdb=> INSERT INTO greeting VALUES ('Привет, мир!');

INSERT 0 1
```

Uzoq masofadan ulanish

Bizning misolimizda mijoz va MBBT bitta kompyuterda. Albatta, PostgreSQLni maxsus serverga o'rnatishingiz va unga boshqa mashinadan (masalan, dastur serveridan) ulanishingiz mumkin. Bunday holda, localhost o'rniga MBBT joylashgan

serveringiz manzilini ko'rsatishingiz kerak. Ammo bu yetarli emas: o'zgarishlarsiz, xavfsizlik nuqtai nazaridan PostgreSQL faqat mahalliy ulanishlarga ruxsat beradi.

Ma'lumotlar bazasiga tashqi ulanishni amalga oshirish uchun siz ikkita faylni tahrirlashingiz kerak.

Birinchidan, postgresql.conf – bu **asosiy sozlamalar fayli** (odatda ma'lumotlar bazasi katalogida joylashgan).

PostgreSQL qaysi tarmoq interfeyslarini tinglashini ko'rsatadigan qatorni toping:

```
#listen_addresses = 'localhost'
```

va uni quyidagiga almashtiring:

```
listen_addresses = '*'
```

Ikkinchidan, pg_hba.conf – bu **autentifikatsiya sozlamalariga ega fayl**.

Mijoz serverga ulanganda, ushbu faylda to'rtta parametrda ulanishga mos keladigan yuqoridan birinchi qator tanlanadi: ulanish turi, ma'lumotlar bazasi nomi, foydalanuvchi nomi va mijozning IP-manzili. Xuddi shu qatorda foydalanuvchi o'zini tanitganda haqiqatdan ham aynan u ekanligini qanday tasdiqlashi kerakligini ko'rsatiladi.

Masalan, Debian va Ubuntuda ushbu fayl, boshqalaridan tashqari holda, quyidagi sozlamani o'z ichiga oladi (xesh bilan boshlangan yuqori qator sharh hisoblanadi):

```
# TYPE  DATABASE  USER  ADDRESS       METHOD
local   all       all                 peer
```

Unda aytilishicha, har qanday foydalanuvchi (`all`) nomidan har qanday ma'lumotlar bazasiga (`all`) mahalliy ulanishlar (`local`) `peer` usuli yordamida tekshirilishi kerak (mahalliy ulanishlar uchun IP-manzil, albatta, ko'rsatilmaydi).

`peer` usuli PostgreSQL operatsion tizimdan joriy foydalanuvchi nomini so'rashini anglatadi va OT allaqachon kerakli tekshiruvni amalga oshirgan deb taxmin qiladi (u foydalanuvchidan parol so'ragan deb biladi). Shuning uchun, Linuxga o'xshash operatsion tizimlarda mahalliy serverga ulanishda odatda parolni kiritish talab qilinmaydi.

Ammo Windowsda mahalliy ulanishlar qo'llab-quvvatlanmaydi va u yerda sozlamalar quyidagicha ko'rinadi:

```
# TYPE  DATABASE  USER  ADDRESS       METHOD
host    all       all   127.0.0.1/32  md5
```

Bu shuni anglatadiki, mahalliy manzildan (`127.0.0.1`) istalgan foydalanuvchi (`all`) nomidan istalgan ma'lumotlar bazasiga (`all`) tarmoq ulanishlari (`host`) `md5` usuli yordamida tekshirilishi kerak. Ushbu usul foydalanuvchidan parolni kiritishni talab qiladi.

To'g'ri parolni ko'rsatishda `app` foydalanuvchisiga `appdb` ma'lumotlar bazasiga istalgan manzildan kirish huquqini berish uchun `pg_hba.conf` oxiriga quyidagi qatorni qo'shing:

```
host    appdb     app   all           md5
```

Konfiguratsiya fayllariga o'zgartirishlar kiritilgandan so'ng, serverdan sozlamalarni qayta o'qishni so'rashni unutmang:

```
postgres=# SELECT pg_reload_conf();
```

Autentifikatsiya sozlamalari haqida batafsil ma'lumot: postgrespro.ru/doc/client-authentication.

Aloqani tekshirish

Dasturingizdan PostgreSQLga ulanish uchun mos kutubxonani toping va MBBT drayverini o'rnating. Drayver odatda PostgreSQL mijoz-server protokolining standart ilovasi bo'lgan `libpq` – uchun qoplama hisoblanadi, ammo boshqa variantlari ham mavjud. Kutubxona dastur ishlab chiquvchilarga past darajadagi protokol imkoniyatlaridan qulay foydalanish imkonini beradi.

Quyida bir nechta mashhur dasturlash tillarida oddiy kod misollari keltirilgan. Ular o'rnatilgan va sozlangan ma'lumotlar bazasiga ulanishni tezda tekshirishga yordam beradi.

Yuqoridagi dasturlar oddiy ma'lumotlar bazasi so'rovini bajarish va natijani chiqarish uchun faqat minimal zarur kodni o'z ichiga oladi. Ular bundan tashqari hech narsa bermaydilar, jumladan, xatolarni qayta ishlash ham inobatga olinmagan. Ushbu kod parchalarini ta'qib qilish uchun misol sifatida qabul qilmaslik kerak.

Windows ostida turli alifbolar belgilarini to'g'ri ko'rsatish uchun buyruq satri oynasida shriftni TrueTypega o'zgartirishingiz kerak bo'lishi mumkin (masalan, "Lucida Console" yoki "Consolas") va shu orqali kod sahifasini ozgartirasiz. Shunday qilib, rus tilida chiqish uchun quyidagi buyruqlarni bering:

```
C:\> chcp 1251

Active code page: 1251

C:\> set PGCLIENTENCODING=WIN1251
```

PHP

PHP tilida PostgreSQL bilan ishlash maxsus kengaytma yordamida tashkil etiladi. Linux ostida, PHPdan tashqari, sizga ushbu kengaytma paketi kerak bo'ladi:

```
$ sudo apt-get install php-cli php-pgsql
```

Windows uchun PHP windows.php.net/download saytida mavjud. PostgreSQL kengaytmasi allaqachon kiritilgan, ammo php.ini faylida siz quyidagi qatorni topishingiz va izoh belgisini olib tashlashingiz (nuqtali vergulni olib tashlashingiz) kerak:

```
;extension=php_pgsql.dll
```

Dastur misoli (test.php):

```
<?php
  $conn = pg_connect('host=localhost port=5432 ' .
                     'dbname=appdb user=app ' .
                     'password=p@ssw0rd') or die;
  $query = pg_query('SELECT * FROM greeting') or die;
  while ($row = pg_fetch_array($query)) {
    echo $row[0].PHP_EOL;
  }
  pg_free_result($query);
  pg_close($conn);
?>
```

Bajaramiz:

```
$ php test.php

Привет, мир!
```

PostgreSQL kengaytmasi ushbu hujjatlarda tasvirlangan: php.net/manual/ru/book.pgsql.php.

Perl

Perlda ma'lumotlar bazalari bilan ishlash DBI interfeysi yordamida tashkil etiladi. Perlning o'zi Debian va Ubuntuda oldindan o'rnatilgan, shuning uchun sizga qo'shimcha ravishda drayver kerak bo'ladi:

```
$ sudo apt-get install libdbd-pg-perl
```

Windows uchun bir nechta Perl yig'ma paketlari mavjud, ular perl.org/get.html havolada keltirilgan. Mashhur ActiveState Perl va Strawberry Perl yig'ma paketlari allaqachon PostgreSQL uchun zarur bo'lgan drayverni o'z ichiga oladi.

Na'muna dastur (test.pl):

```
use DBI;
use open ':std', ':utf8';
my $conn = DBI->connect(
  'dbi:Pg:dbname=appdb;host=localhost;port=5432',
  'app',
  'p@ssw0rd') or die;
my $query = $conn->prepare('SELECT * FROM greeting');
$query->execute() or die;
while (my @row = $query->fetchrow_array()) {
  print @row[0]."\n";
}
$query->finish();
$conn->disconnect();
```

Bajaramiz:

```
$ perl test.pl

Привет, мир!
```

Interfeys hujjatlarda tasvirlangan:
metacpan.org/pod/DBD::Pg.

Python

Python odatda PostgreSQL bilan ishlash uchun psycopg kutubxonasidan (“syco-pee-gee” deb talaffuz qilinadi) foydalanadi.

Debian va Ubuntuning zamonaviy talqinlari oldindan o'rnatilgan Python 3-talqini bilan birga keladi, shuning uchun sizga faqat drayver kerak bo'ladi:

```
$ sudo apt-get install python3-psycopg2
```

Windows operatsion tizimi uchun Pythonni python.org saytidan olish mumkin. Psycopg kutubxonasi initd.org/psycopg loyiha veb-saytida mavjud (o'rnatilgan Python versiyasiga mos keladigan versiyani tanlang).

Kerakli hujjatlar ham u yerda joylashgan. Na'muna dastur (test.py):

```
import psycopg2
conn = psycopg2.connect(
  host='localhost',
  port='5432',
  database='appdb',
  user='app',
  password='p@ssw0rd')
cur = conn.cursor()
cur.execute('SELECT * FROM greeting')
rows = cur.fetchall()
for row in rows:
  print(row[0])
conn.close()
```

Bajaramiz:

```
$ python3 test.py

Привет, мир!
```

Java

Java tilida ma'lumotlar bazalari bilan ishlash JDBC interfeysi orqali tashkil etiladi. Java SE 11ni o'rnating; Bundan tashqari, bizga JDBC drayveri bo'lgan paket kerak:

```
$ sudo apt-get install openjdk-11-jdk
```

```
$ sudo apt-get install libpostgresql-jdbc-java
```

Windows OT uchun JDKni oracle.com/technetwork/java/javase/downloads havoladan yuklab olish mumkin. JDBC drayveri jdbc.postgresql.org saytida mavjud (siz o'rnatgan JDK talqiniga mos keladigan talqinni tanlang). Hujjatlar ham u yerda joylashgan.

Na'muna dastur (Test.java):

```
import java.sql.*;
public class Test {
  public static void main(String[] args)
  throws SQLException {
    Connection conn = DriverManager.getConnection(
      "jdbc:postgresql://localhost:5432/appdb",
      "app", "p@ssw0rd");
    Statement st = conn.createStatement();
    ResultSet rs = st.executeQuery(
      "SELECT * FROM greeting");
    while (rs.next()) {
      System.out.println(rs.getString(1));
    }
    rs.close();
    st.close();
    conn.close();
  }
}
```

Biz dasturni kompilyatsiya qilamiz va bajaramiz, kalitda JDBC drayverlari sinfiga yo'lni ko'rsatamiz (Windowsda yo'llar ikki nuqta bilan emas, balki nuqta-vergul bilan ajratiladi):

```
$ javac Test.java
$ java -cp .:/usr/share/java/postgresql-jdbc4.jar \
Test
Привет, мир!
```

Zaxira uchun nusxa yaratish

Biz yaratgan ma'lumotlar bazasida faqat bitta jadval mavjud bo'lsada, ma'lumotlar xavfsizligi haqida o'ylash zarar qilmaydi. Ilovada ko'p ma'lumotlar bo'lmasada, zaxira uchun nusxa yaratishning eng oson yo'li pg_dump yordamchi dasturidir:

```
$ pg_dump appdb > appdb.dump
```

Olingan appdb.dump fayli barcha appdb obyektlarini yaratadigan va ma'lumotlar bilan to'ldiruvchi oddiy SQL buyruqlarini o'z ichiga oladi. Ushbu fayl psql uchun argument sifatida berilib va ma'lumotlar bazasi tarkibini tiklash mumkin. Masalan, siz yangi ma'lumotlar bazasini yaratishingiz va unga ma'lumotlarni yuklashingiz mumkin:

```
$ createdb appdb2
$ psql -d appdb2 -f appdb.dump
```

Aynan shu shaklda biz oldingi bobda tanishgan demo bazasi tarqatiladi.

pg_dumpda tanishishga arziydigan ko'plab xususiyatlar mavjud: postgrespro.ru/doc/app-pgdump. Ulardan ba'zilari faqat ma'lumotlar maxsus ichki formatda yuklanganda mavjud; bunday holda, tiklash uchun siz psql emas, balki pg_restore yordamchi dasturidan foydalanishingiz kerak.

Ikkala holatda ham pg_dump faqat bitta ma'lumotlar bazasi tarkibini o'chiradi. Agar siz barcha ma'lumotlar bazalari, foydalanuvchilar va jadval maydonlarini o'z ichiga olgan klasterning zaxira nusxasini yaratmoqchi bo'lsangiz, boshqa, shunga o'xshash bo'lgan pg_dumpall buyrug'idan foydalanishingiz kerak.

Katta, jiddiy loyihalar o'ylangan davriy zaxira strategiyasini talab qiladi. Buning uchun pg_basebackup yordamchi dasturi tomonidan yaratilgan klasterning fizik "ikkilik" nusxasi ko'proq mos keladi:

```
$ pg_basebackup -D backup
```

Ushbu buyruq backup katalogidagi barcha ma'lumotlar bazasi klasterining zaxira nusxasini yaratadi. Tizimni yaratilgan nusxadan tiklash uchun uni ma'lumotlar bazasi katalogiga nusxalash va serverni ishga tushirish kifoya.

Hujjatlarda mavjud bo'lgan barcha zaxira va tiklash vositalari haqida ko'proq ma'lumot olishingiz mumkin: postgrespro.ru/doc/backup, shuningdek, DBA3 o'quv kursidan ham foydalanishingiz mumkin (161-bet).

Standart PostgreSQL vositalari sizga kerak bo'lgan deyarli hamma narsani bajarishga imkon beradi, lekin avtomatlashtirishni talab qiladigan ko'plab qadamlarni bajarish kerak bo'ladi. Shuning uchun ko'pgina kompaniyalar o'zlarining zaxira va tiklash vositalarini yaratadilar. Bizning Postgres Professional kompaniyamiz ham shunday vositaga ega – pg_probackup. Asbobning bepul talqini sahifa darajasida bosqichma-bosqich zaxira nusxalarini yaratish, ma'lumotlar yaxlitligini kuzatish, parallellik va siqish orqali katta hajmdagi ma'lumotlar bilan ishlash va turli xil zaxira strategiyalarini amalga oshirish imkonini beradi. To'liq hujjatlar postgrespro.ru/doc/app-pgprobackup saytida mavjud.

Keyin nima?

Endi siz ilovangizni ishlab chiqishga tayyorsiz. Ma'lumotlar bazasiga nisbatan u har doim ikki qismdan iborat bo'ladi: server va mijoz. Server qismi ma'lumotlar bazasi bilan bog'liq bo'lgan barcha narsalar: jadvallar, indekslar, ko'rinishlar, triggerlar, saqlangan funktsiyalar va protseduralar. Mijoz – ma'lumotlar bazasidan tashqarida ishlaydigan va unga ulanadigan hamma narsa. Ma'lumotlar bazasi nuqtai nazaridan, u qalin mijoz yoki dastur serveri bo'ladimi, muhim emas.

Aniq to'g'ri javob bo'lmagan muhim savol: dasturning biznes mantig'i qayerda to'planishi kerak?

Ommabop yondashuv – bu barcha mantiq mijozda, ma'lumotlar bazasidan tashqarida yotishi. Bu ko'proq ishlab chiquvchilar relyatsion MBBTning barcha imkoniyatlarini bilmasa va ular faqat amaliy kod yozishni bilishgandagina, ya'ni dastur kodi bilan ishlaganda tanlanadi.

Bunday holda, ma'lumotlar bazasi dasturning o'ziga xos ikkilamchi elementiga aylanadi va faqat ma'lumotlarning "qat'iyligi" va uning ishonchli saqlanishini ta'minlaydi. Ko'pincha, ma'lumotlar bazasi ma'lumotlar bazasi qo'shimcha abstraktsiya qatlami bilan o'ralgan, masalan, ishlab chiquvchilarga tanish bo'lgan dasturlash tilidagi konstruktsiyalardan ma'lumotlar bazasiga so'rovlarni avtomatik ravishda ishlab chiqaradigan ORM. Ba'zan bu qaror ilovaning har qanday ma'lumotlar bazasiga ko'chirilishini ta'minlash istagi bilan izohlanadi.

Yondashuv mavjud bo'lish huquqiga ega va shu tarzda qurilgan tizim ishlasa hamda unga yuklatilgan vazifalarni bajarsa, nega endi yo'q?

Ammo bu yechimning aniq kamchiliklari ham bor:

- **Ma'lumotlar izchilligi ilova tomonidan ta'minlanadi.** Ma'lumotlar bazasiga ma'lumotlarning izchilligini ta'minlashni ishonib topshirish o'rniga (bu aynan relyatsion tizimlarning kuchli deb qaraladigan tomoni), dastur o'z-o'zidan kerakli tekshiruvlarni amalga oshiradi. Ertami-kechmi noto'g'ri ma'lumotlar ma'lumotlar bazasiga tushishi aniq. Ushbu xatolarni tuzatish kerak bo'ladi – yoki dastur ular bilan ishlashga o'rgatilgan bo'lishi kerak. Ammo bir ma'lumotlar bazasida bir nechta turli xil ilovalar yaratilgan holatlar mavjud. Bunday holda, MBBT yordamisiz buni amalga oshirish mumkin emas.
- **Ishlab turishi yanada yaxshi ishlashga umid qoldiradi.** ORM tizimlari sizga ma'lum darajada ma'lumotlar bazasidan ma'lumot olish imkonini beradi, ammo ular yaratadigan SQL so'rovlarining sifati juda shubhali. Odatda, ko'plab kichik so'rovlar bajariladi, ularning har biri o'z-o'zidan juda tezdir. Ushbu sxema faqat kichik hajmdagi ma'lumotlarga kichik yuklamalarga bardosh bera oladi va MBBT tomonidan hech qanday optimallashtirishga amalda mos kelmaydi.
- **Ilova kodi yanada murakkablashadi.** Avtomatik va adekvat ravishda SQLga tarjima qilinadigan amaliy dasturlash tilida chinakam murakkab so'rovni shakllantirish mumkin emas. Shuning uchun murakkab ishlov berish (agar kerak bo'lsa, albatta) dastur darajasida dasturlashtirilgan bo'lishi kerak, birinchi navbatda ma'lumotlar bazasidan barcha kerakli ma'lumotlarni tanlab olish kerak. Lekin, birinchidan, bu tarmoq orqali ma'lumotlarni keraksiz uzatishni o'z ichiga oladi, ikkinchidan, ma'lumotlar bazasida skanerlash, qo'shilish, saralash va yig'ish kabi algoritmlar o'n yillar davomida diskka moslab qilingan, optimallashtirilgan va dastur kodidan ko'ra yaxshiroq vazifani bajarishi kafolatlangan.

Albatta, barcha yaxlitlik cheklovlarini va saqlangan funksiyalarda ma‘lumotlar bilan ishlash mantig‘ini amalga oshirish bilan ma‘lumotlar bazasidan to‘liq foydalanish uning xususiyatlari va imkoniyatlarini chuqur o‘rganishni talab qiladi. So‘rovlarni yozish uchun SQL tilini va funksiyalar va triggerlarni yozish uchun ba‘zi server dasturlash tilini (odatda PL/pgSQL) o‘zlashtirishingiz kerak bo‘ladi. Buning evaziga siz har qanday axborot tizimi arxitekturasining muhim “qurilish bloklaridan” biri bo‘lgan ishonchli vositani o‘zlashtirasiz.

Qanday bo‘lmasin, biznes mantig‘ini qayerda – serverda yoki mijozda joylashtirish masalasini o‘zingiz hal qilishingiz kerak bo‘ladi. Faqat shuni qo‘shimcha qilaylikki, haddan tashqari narsalar har doim ham zarur emas va ko‘pincha haqiqatni o‘rtada qidirish kerak.

VII PostgreSQLni sozlash

Asosiy sozlashlar

PostgreSQL sukut bo'yicha, server hatto eng zaif uskunada ham ishlashi uchun sozlangan, ammo MBBT samarali ishlashi uchun sozlashda siz serverning jismoniy xususiyatlarini ham, dasturni yuklamasini ham hisobga olishingiz kerak.

Bu yerda biz faqat ma'lumotlar bazasini haqiqiy ekspluatatsiyasida alohida e'tibor talab qiladigan asosiy sozlamalarni ko'rib chiqamiz. Muayyan dastur uchun nozik sozlash qo'shimcha bilimlarni talab qiladi, masalan, PostgreSQLni boshqaruv kurslarini o'tish orqali olish mumkin (155-betga qarang).

Konfiguratsiya parametrlarini qanday o'zgartirish mumkin

Konfiguratsiya parametrining qiymatini o'gartirish uchun postgresql.conf faylini oching va kerakli parametrni toping va uning qiymatini o'zgartiring yoki faylning oxiriga yangi qator qo'shing – u avval aynan shu faylda o'rnatilgan qiymatdan ustun bo'ladi.

Keyin serverga sozlamalarni qayta o'qishni ayting:

```
postgres=# SELECT pg_reload_conf();
```

Shundan so'ng, SHOW buyrug'i yordamida parametrning joriy qiymatini tekshiring. Agar qiymat o'zgarmagan bo'lsa, faylni tahrirlashda xatolik yuz bergan bo'ladi. Server xabarlari jurnaliga qarang.

Faylni matn muharririda o'zgartirish o'rniga, parametr qiymatini SQL yordamida o'rnatishingiz mumkin (bundan keyin siz sozlamalarni qayta o'qib chiqishingiz kerak):

```
postgres=# ALTER SYSTEM SET work_mem='128MB';
```

Bunday sozlamalar postgresql.auto.conf fayliga kiradi va asosiy faylda o'rnatilgan qiymatlardan ustun turadi. Ushbu usulning afzalligi shundaki, parametr qiymatlari to'g'riligi darhol tekshiriladi.

Muhimroq parametrlar

Eng muhimlaridan ba'zilari – PostgreSQL RAMni qanday boshqarishini aniqlaydigan parametrlardir.

shared_buffers parametri bufer keshining kerakli hajmini o'rnatadi, shuning uchun tez-tez ishlatiladigan ma'lumotlar bilan ishlash RAMda amalga oshiriladi va ortiqcha diskdan foydalanishni talab qilmaydi. Serverning jami operativ xotirasining 25% bilan sozlashni boshlashingiz mumkin. Ushbu parametrni o'zgartirish faqat server qayta ishga tushirilgandan so'ng kuchga kiradi!

effective_cache_size parametrining qiymati xotira taqsimotiga ta'sir qilmaydi, lekin PostgreSQLga qanday umumiy kesh hajmini, shu jumladan operatsion tizim keshini ham inobatga olgan holda ishlatishni maqsad qilish kerakligini aytadi. Bu

qiymat qanchalik yuqori bo'lsa, indekslarga ko'proq ustunlik beriladi. Siz RAMning 50–75% idan boshlashingiz mumkin.

work_mem parametri birlashma (join) amalga oshirilganda tartiblash yoki xesh jadvallarini yaratish kabi operatsiyalarni bajarish uchun ajratilgan xotira hajmini aniqlaydi. Xotira yetarli emasligining belgisi vaqtinchalik fayllardan faol foydalanish va natijada ishlashning pasayishi hisoblanadi. Ko'p hollarda 4 MB standart qiymati kamida bir necha marta oshirilishi kerak, lekin serverning operativ xotirasining umumiy hajmidan oshmasligi kerak.

maintenance_work_mem parametri xizmat jarayonlariga ajratilgan xotira hajmini aniqlaydi. Uni oshirish indeks yaratish va vakuum jarayonini tezlashtirishi mumkin. Odatda *work_mem* qiymatidan bir necha marta kattaroq qiymatga moslab o'rnatiladi.

Masalan, 32 GB operativ xotira bilan siz quyidagilarni sozlashdan boshlashingiz mumkin:

```
shared_buffers = '8GB'
effective_cache_size = '24GB'
work_mem = '128MB'
maintenance_work_mem = '512MB'
```

random_page_cost va *seq_page_cost* ikkita parametr qiymatlarining nisbati tasodifiy va ketma-ket kirish tezligi nisbatiga mos kelishi kerak. Odatda, tasodifiy kirish ketma-ket kirishdan 4 baravar sekinroq (oddiy HDDlarda hisoblangan) deb hisoblanadi. Biroq, disk massivlari hamda SSD qurilmalar uchun ***random_page_cost*** qiymatini kamaytirish kerak (shunchaki hech qachon *seq_page_cost* qiymatini 1ga o'zgartirmang).

Masalan, SSD drayvlar uchun quyidagi sozlama yetarli bo'ladi:

```
random_page_cost = 1.2
```

Avtotozalashni o'rnatish (autovacuum) juda muhimdir. Bu jarayon "axlat yig'ish" bilan shug'ullanadi va bir qator boshqa tizim uchun muhim vazifalarni bajaradi. Uning konfiguratsiyasi sezilarli darajada dastur va u yaratadigan yuklamaga bog'liq.

Ko'pgina hollarda siz quyidagilardan boshlashingiz mumkin:

- ***autovacuum_vacuum_scale_factor*** qiymatini 0,01ga kamaytirish orqali tozalashni tez-tez va kichikroq qismlarda amalga oshirilishini ta'minlang;
- ***autovacuum_vacuum_cost_limit*** qiymatini 10 barobarga oshirib (yoki ***autovacuum_vacuum_cost_delay***ni kamaytirib) (12 chigacha bo'lgan talqinlar uchun) tozalashni tezroq bajarilishini ta'minlang.

Bufer keshini va oldindan yozish jurnalini saqlash bilan bog'liq jarayonlarni o'rnatish bundan kam ahamiyatga ega emas, lekin u ham muayyan dasturga bog'liq. Yuklamani yengillashtirish uchun ***checkpoint_completion_target*** = 0,9ni o'rnatishdan boshlang, `checkpoint_timeout` vaqtini 5 daqiqadan 30ga oshirish (nazorat punktiga qo'shimcha xarajatlarni kamaytirish uchun) va ***max_wal_size***ni proporsional ravishda oshirish (xuddi shu maqsadda).

Turli parametrlarni o'rnatishning nozik jihatlari DBA2 o'quv kursida batafsil muhokama qilinadi (160-bet).

Ulanishni sozlash

Biz bu masalani 95 betdagi "Ilova uchun PostgreSQL" bobida muhokama qilganmiz. Eslatib o'tamiz, odatda

listen_addresses parametrini '*' ga o'rnatishingiz va pg_hba.conf konfiguratsiya fayliga ulanish ruxsatini qo'shishingiz kerak.

Yomon maslahatlar

Siz hech qachon tinglamasligingiz kerak bo'lgan samaradorlikni oshirish bo'yicha maslahatlarni topishingiz mumkin:

- Avtomatik tozalashni o'chiring (*autovacuum* = off).

  Resurslarni bunday "tejash" haqiqatan ham qisqa muddatli samaradorlikni beradi, lekin ma'lumotlarda "axlat" to'planishiga va jadvallar va indekslarning o'sishiga olib keladi. Biroz vaqt o'tgach, MBBT normal ishlashini to'xtatadi. Avtomatik tozalashni o'chirib qo'ymaslik kerak, lekin to'g'ri sozlangan bo'lishi kerak.

- Disk sinxronizatsiyasini o'chiring (*fsync* = off).

  O'chirish haqiqatan ham sezilarli tezlikka olib keladi, lekin serverning har qanday ishlamay qolishi (dasturiy ta'minot yoki apparat) ma'lumotlar bazalarining to'liq yo'qolishiga olib keladi. Tizimni faqat zaxira nusxasidan tiklash mumkin bo'ladi (agar, albatta, mavjud bo'lsa).

PostgreSQL va 1C

1C rasmiy ravishda PostgreSQL bilan ishlashni qo'llab-quvvatlaydi. Bu tijorat MBBTlar uchun qimmat litsenziyalarni tejash uchun ajoyib imkoniyatdir.

Har qanday boshqa ilovalar singari, PostgreSQL to'g'ri sozlangan bo'lsa, 1C mahsulotlari ham ancha samarali ishlaydi.

Bundan tashqari, 1C ishlashi uchun o'ziga xos va zarur bo'lgan bir qator parametrlar mavjud.

Quyida tezda boshlashingizga yordam beradigan o'rnatish va dastlabki sozlash ko'rsatmalari keltirilgan.

Talqin va platformalarni tanlash

1C bilan ishlash uchun sizga PostgreSQLning maxsus patchli talqini kerak bo'ladi. Ushbu talqinni 1C releases.1c.ru veb-saytidan olish mumkin yoki siz Postgres Pro Standard yohud Postgres Pro Enterprise MBBTdan foydalanishingiz mumkin, ular ham kerakli patchlarni o'z ichiga oladi.

PostgreSQL Windows tizimida ham ishlaydi, ammo tanlov imkoniyati bo'lsa, Linux OTga ustunlik berishingiz kerak.

O'rnatishdan oldin siz ma'lumotlar bazasi uchun maxsus server kerak yoki yo'qligini hal qilishingiz kerak. Ajratilgan server yuklamani dastur serveri va ma'lumotlar bazasi serveri o'rtasida taqsimlash orqali samaraliroq bo'ladi.

Konfiguratsiya parametrlari

Serverning jismoniy xususiyatlari kutilgan yuklamaga mos kelishi kerak. Quyidagi ma'lumotlar taxminiy ko'rsatma sifatida berilishi mumkin. 8 Gb tezkor xotiraga ega 8 yadroli va RAID1 SSD diskga ega maxsus server 100 Gb gacha bo'lgan ma'lumotlar bazasi hajmiga, 50 kishigacha bo'lgan umumiy foydalanuvchilar soniga va kunlik 2000 tagacha hujjatlarga ega bardosh bera olishi kerak. Agar server ajratilmagan bo'lsa, PostgreSQL ehtiyojlari uchun umumiy server resurslarining tegishli miqdori ajratilgan bo'lishi kerak.

Yuqorida keltirilgan umumiy tavsiyalar va 1C ilovalarining o‘ziga xos xususiyatlariga asoslanib, bunday server uchun biz quyidagi dastlabki sozlamalarni tavsiya qilamiz:

```
# 1C uchun zarur sozlashlar
standard_conforming_strings = off
escape_string_warning = off
shared_preload_libraries = 'online_analyze, plantuner'
plantuner.fix_empty_table = on
online_analyze.enable = on
online_analyze.table_type = 'temporary'
online_analyze.local_tracking = on
online_analyze.verbose = off


# Tezkor xotiraga bog‘liq parametrlar
shared_buffers = '2GB'          # 25% RAM
effective_cache_size = '6GB'    # 75% RAM
work_mem = '64MB'               # 64-128MB
maintenance_work_mem = '256MB' # 4*work_mem
# vaqtinchalik jadvallar bilan faol ishlaganda
temp_buffers = '32MB'           # 32-128MB


# oz‘garishsiz 64 bo‘lgan qiymatdan ko‘ra ko‘proq
# blocklashlar talab etilganda
max_locks_per_transaction = 256
```

Ulanishni sozlash

postgresql.conf faylidagi *listen_addresses* parametri `'*'`ga o'rnatilishi kerak.

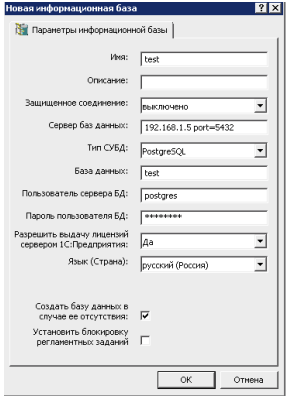
Siz pg_hba.conf konfiguratsiya faylining boshiga quyidagi qatorni qo'shishingiz kerak, “1C server IP manzili”ni ma'lum bir manzil va qism tarmoq maskasi bilan almashtiring:

```
host    all    all    IP-адрес-сервера-1C   md5
```

PostgreSQLni qayta ishga tushirgandan so'ng, postgresql.conf va pg_hba.conf fayllaridagi barcha o'zgarishlar kuchga kiradi va server 1Cga ulanishga tayyor bo'ladi.

Ulanish uchun 1C superuser rolidan foydalanadi, odatda `postgres`. Uning uchun parol o'rnating:

```
postgres=# ALTER ROLE postgres PASSWORD 'p@ssw0rd';
ALTER ROLE
```



1C ma'lumotlar bazasi sozlamalarida ma'lumotlar bazasi serveri sifatida PostgreSQL serverining IP manzili va portini belgilang va "PostgreSQL" MBBT turini tanlang. Foydalanadigan ma'lumotlar bazasi nomini belgilang 1C-ni oching va "Создать базу данных в случае ее отсутствия" katagiga belgi qo'ying (PostgreSQL yordamida ma'lumotlar bazasini yaratishning hojati yo'q). Ulanish uchun foydalaniladigan superuser rolining nomi va parolini belgilang.

Yuqoridagi maslahatlar sizni ishni tezda boshlashingiz uchun yetarli, albatta, ular kerakli darajada ishlashni kafolatlamaydi.

"Инфософт" kompaniyasidan Anton Doroshkevichga ushbu materialni tayyorlashda yordam bergani uchun o'z minnatdorchiligimizni bildiramiz.

VIII pgAdmin

pgAdmin – PostgreSQL boshqaruvi uchun mashhur grafik muhit. Dastur asosiy boshqaruv vazifalarini soddalashtiradi, ma'lumotlar bazasi obyektlarini ko'rsatadi va SQL so'rovlarini bajarishga imkon beradi.

Uzoq vaqt davomida pgAdmin 3 de-fakto standart edi, lekin EnterpriseDB ishlab chiquvchilari uni qo'llab-quvvatlashni to'xtatdilar va 2016 yilda mahsulotni C++ tilidan Python va veb-texnologiyalarga to'liq qayta yozib, yangi, to'rtinchi talqinni chiqardilar. O'zgartirilgan interfeys tufayli pgAdmin 4 dastlab juda sovuqqonlik bilan qabul qilindi, ammo ishlab chiqilish va takomillashtirishda davom etmoqda.

O'rnatish

Windows tizimida pgAdmin 4ni ishga tushirish uchun pgadmin.org/download manzilidagi o'rnatuvchidan foydalaning. O'rnatish jarayoni oddiy va tushunarli, barcha tavsiya etilgan qiymatlar o'zgarishsiz qoldirilishi mumkin.

Debian va Ubuntu uchun PostgreSQL bazasini ulang (34-betda tasvirlanganidek) va quyidagi buyruqni amalga oshiring

```
$ sudo apt-get install pgadmin4
```

Mavjud dasturlar ro'yxatida "pgAdmin4" paydo bo'ladi.

Foydalanuvchi interfeysi tilini rus tiliga o'zgartirish uchun **Настроить pgAdmin** (Configure pgAdmin) belgisini bosing va sozlamalar oynasida **Разное > Язык пользователя** (Miscellaneous > User language) tanlang. So'ngra veb-brauzeringizda sahifani qayta yuklang.

Serverga ulanish

Avvalo, serverga ulanishni sozlaymiz. **Добавить новый сервер** – Yangi server qo'shish (Add New Server) belgisini bosing va paydo bo'lgan oynaning **Общие** – Umumiy (General) taxlamida ulanish uchun ixtiyoriy **имя** – nom (Name)ni kiriting.

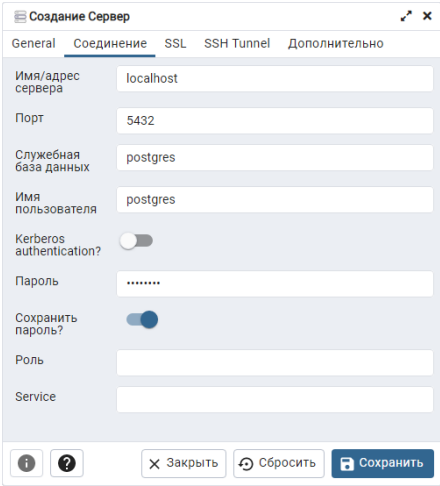
Соединение (Connection) taxlamida **имя сервера** (Host name/address) – server nomi, **порт** (Port) – port, **имя пользователя** (Username) – foydalanuvchi nomi va **пароль** (Password) – parolni kiriting.

Agar siz har safar parolingizni qo'lda kiritishni xohlamasangiz, **Сохранить пароль** (Save password) – "Parolni saqlash" katagiga belgi qo'ying. Parollar asosiy parol yordamida shifrlangan holda saqlanadi, pgAdmin uni birinchi ishga tushirganingizda o'rnatishingizni so'raydi.

E'tibor bering, foydalanuvchi paroli o'rnatilgan bo'lishi kerak. Masalan, `postgres` uchun bu quyidagi buyruq bilan amalga oshirilishi mumkin:

```
postgres=# ALTER ROLE postgres PASSWORD 'p@ssw0rd';
```

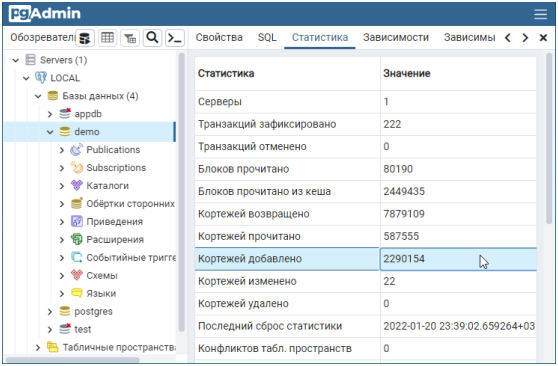
Сохранить (Save) – saqlash tugmasini bosganingizda dastur belgilangan parametrlar bilan server mavjudligini tekshiradi va yangi ulanishni eslab qoladi.



Navigator

Oynaning chap tomonida obyekt navigatori joylashgan. Ro'yxat elementlarini kengaytirib, biz LOCAL deb atagan serverga o'tishingiz mumkin. Quyida unda mavjud bo'lgan ma'lumotlar bazalari ro'yxati keltirilgan:

- `appdb` biz turli dasturlash tillaridan PostgreSQLga ulanishlarni sinab ko'rish uchun yaratdik;
- `demo` – demo ma'lumotlar bazasi;
- `postgres` MBBTni o'rnatishda doimo yaratiladi;
- `test` SQL bilan tanishganda ishlatdik.



demo ma'lumotlar bazasi uchun **Схемы** (Schemas) – "Sxemalar" elementini ochish orqali siz barcha jadvallarni topishingiz, ularning ustunlarini, yaxlitlik cheklovlarini, indekslarni, triggerlarni va boshqalarni ko'rishingiz mumkin.

Har bir turdagi obyekt uchun kontekst menyusi (sichqonchaning o'ng tugmasi) u bilan bajarilishi mumkin bo'lgan harakatlar ro'yxatini o'z ichiga oladi. Masalan, faylga yuklash yoki fayldan yuklash, imtiyozlar berish, o'chirish.

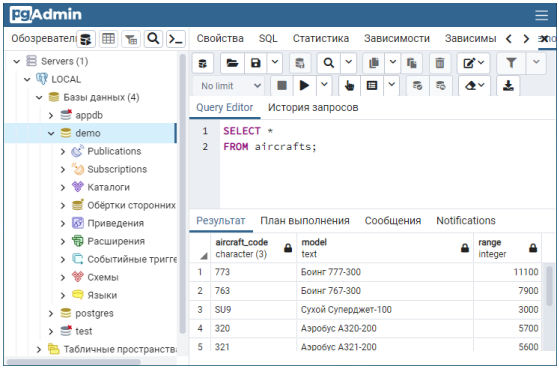
Oynaning o'ng tomonida ma'lumotnoma ma'lumotlari alohida yorliqlarda ko'rsatiladi:

- **Панель информации** (Dashboard) – Axborot paneli – tizim faoliyatini aks ettiruvchi grafiklarni ko'rsatadi;
- **Свойства** (Properties) – xossalar – tanlangan obyektning xossalari (ustun uchun uning ma'lumotlar turi va hokazolar ko'rsatiladi);

- **SQL** – navigatorda tanlangan obyektni yaratish uchun SQL buyrug‘i;
- **Статистика** (Statistics) – statistika – optimallashtiruvchi tomonidan so‘rovlarni bajarish rejalarini tuzish uchun foydalaniladigan va vaziyatni tahlil qilish uchun ma‘lumotlar bazasi ma‘muri tomonidan ko‘rilishi mumkin bo‘lgan ma‘lumotlar;
- **Зависимости**, **Зависимые** (Dependencies, Dependents) – bog‘liqliklar, bog‘liq bo‘lganlar – tanlangan obyekt va ma‘lumotlar bazasidagi boshqa obyektlar o‘rtasidagi bog‘liqlikni ko‘rsatadi.

So‘rovlarni bajarish

So‘rovni bajarish uchun menyudan **Инструменты – Запросник** (Tools – Query tool) – Asboblar – So‘rovlar vositasini tanlash orqali SQL oynasi bilan yangi taxlamni oching.



Oynaning yuqori qismida so'rovingizni kiriting va F5 tugmasini bosing. So'rov natijasi oynaning pastki qismida **Результат** (Data Output) taxlamida paydo bo'ladi.

Yangi so'rovni kiritish uchun oldingisini o'chirish shart emas: kerakli kod qismini ajratib ko'rsatish va F5 tugmasini bosish kifoya. Keyin harakatlaringiz tarixi har doim ko'rinadi – odatda bu **История запросов** (Query History) – "So'rovlar tarixi" yorlig'idagi buyruqlar tarixida kerakli so'rovni qidirishdan ko'ra qulayroqdir.

Boshqa

pgAdmin dasturi standart PostgreSQL yordam dasturlari, tizim katalogi ma'lumotlari, administratorlik funktsiyalari va SQL buyruqlari uchun grafik interfeysni taqdim etadi. Biz, ayniqsa, o'rnatilgan PL/pgSQL kodini tuzatuvchiga e'tibor qaratamiz. Ushbu dasturning barcha xususiyatlarini mahsulotning pgadmin.org veb-saytida yoki dasturning yordam tizimida topish mumkin.

IX Qo‘shimcha imkoniyatlar

Umummatn qidiruv

Umummatnli qidiruv – bu hujjatlarni tabiiy tillarda qidirish, odatda tegishliligi bo‘yicha saralanadi. Eng oddiy va odatiy holatda so‘rov so‘zlar to‘plamiga aylanadi va mos keladigan mezon ularning hujjatdagi uchrash chastotasi hisoblanadi. Google va Yandex shunga o‘xshash tarzda so‘rovlarimiz uchun qidiruvni amalga oshiradi. Biroq, SQL so‘rovlar tilining barcha kuchi ham bunday ma‘lumotlar bilan samarali ishlash uchun yetarli emas. Bu, ayniqsa, so‘nggi paytlarda, axborot bazalari katta ma‘lumotlarning ko‘chkiga o‘xshash oqimi bilan to‘ldirilganida va ularning ko‘pi ko‘pincha yomon tuzilgan va tuzilishini yasash imkoni bo‘lmaganida sezilarli bo‘ldi.

Mavjud bo‘lgan barcha hujjatlarni indekslash va juda yuqori sifatli qidiruvni tashkil qilish imkonini beruvchi pullik va bepul ko‘plab qidiruv tizimlari mavjud. Bunday hollarda indeks – muhim vosita va qidiruv tezlatgichi – ma‘lumotlar bazasining bir qismi emas. Bu shuni anglatadiki, ma‘lumotlar bazasi tarkibini sinxronlashtirish, tranzaksiya, metama‘lumotlarga kirish, ularning yordami bilan qidiruv maydonini cheklash, hujjatlarga kirish siyosatini tartibga solish va boshqa ko‘p

narsalar kabi MBBT foydalanuvchilari tomonidan qadrlanadigan xususiyatlardan foydalanish uchun imkoniyat mavjud bo'lmaydi.

Hujjatga yo'naltirilgan ma'lumotlar bazasining tobora ommalashib borayotgan kamchiliklari odatda bir xil: to'liq matnli qidiruv vositalari ishlab chiqilgan, ammo xavfsizlik va sinxronizatsiya ustuvor ahamiyatga ega emas. Bundan tashqari, ular odatda (masalan, MongoDB) NoSQL MBBT sinfiga kiradi, ya'ni ta'rifiga ko'ra, ular o'nlab yillar davomida to'plangan SQL kuchidan mahrum bo'lishadi.

Boshqa tomondan, an'anaviy SQL MBBTlari o'zlarida o'rnatilgan matn qidirish imkoniyatlariga ega. `LIKE` operatori standart SQL sintaksisining bir qismidir, ammo uning moslashuvchanligi yetarli darajada emas, shu sababli ishlab chiquvchilar SQL standartiga o'zlarining kengaytmalarini qo'shishlari kerak bo'lgan. PostgreSQLda bular `ILIKE`, `~`, `~*` taqqoslash operatorlari, ammo ular ham barcha muammolarni hal qilmaydi, chunki ular so'z o'zgarishini hisobga olmaydi, reyting algoritmi uchun mos kelmaydi va juda tez ishlamaydi.

Agar biz to'liq matnli qidiruv vositalari haqida gapiradigan bo'lsak, ularning standartlashuvi hali ham uzoqda ekanligini tushunish kerak – har bir ma'lumotlar bazasining ularni qo'llash bo'yicha o'z sintaksisi va o'ziga xos yondashuvlari mavjud. Shu munosabat bilan PostgreSQL rus foydalanuvchisi uchun juda qulay: to'liq matnli qidiruv rus ishlab chiquvchilari tomonidan amalga oshiriladi va agar kerak bo'lsa, mutaxassislar bilan bevosita bog'lanish yoki hatto ularning ma'ruzalarida qatnashish orqali uni batafsilroq tushunishingiz mumkin. Biz oddiy misollar bilan cheklanamiz.

Umummatnli qidiruv imkoniyatlarini o'rganish uchun demo ma'lumotlar bazasida boshqa jadval yarataylik. Bular ma'ruza bo'limlariga bo'lingan asosiy eslatmalar bo'lsin:

```
test=# CREATE TABLE course_chapters(
  c_no text REFERENCES courses(c_no),
  ch_no text,
  ch_title text,
  txt text,
  CONSTRAINT pkt_ch PRIMARY KEY(ch_no, c_no)
);

CREATE TABLE
```

Keling, bizga tanish CS301 va CS305 mutaxassisliklari bo'yicha birinchi ma'ruza matnlarini jadvalga kiritamiz:

```
test=# INSERT INTO course_chapters(
  c_no, ch_no,ch_title, txt)
VALUES
('CS301', 'I', 'Базы данных',
 'С этой главы начинается наше знакомство ' ||
 'с увлекательным миром баз данных'),
('CS301', 'II', 'Первые шаги',
 'Продолжаем знакомство с миром баз данных. ' ||
 'Создадим нашу первую текстовую базу данных'),
('CS305', 'I', 'Локальные сети',
 'Здесь начнется наше полное приключений ' ||
 'путешествие в интригующий мир сетей');

INSERT 0 3
```

Natijani tekshiramiz:

```
test=# SELECT ch_no AS no, ch_title, txt
FROM course_chapters \gx

-[ RECORD 1 ]-----------------------------------------
no       | I
ch_title | Базы данных
txt      | С этой главы начинается наше знакомство с
           увлекательным миром баз данных
-[ RECORD 2 ]-----------------------------------------
no       | II
ch_title | Первые шаги
txt      | Продолжаем знакомство с миром баз данных.
           Создадим нашу первую текстовую базу данных
```

```
-[ RECORD 3 ]-------------------------------------
no       | I
ch_title | Локальные сети
txt      | Здесь начнется наше полное приключений
           путешествие в интригующий мир сетей
```

LIKE operatoridan foydalanib, an'anaviy SQL vositalaridan foydalangan holda jadvaldagi ma'lumotlar bazasi ma'lumotlarini topamiz:

```
test=# SELECT txt
FROM course_chapters
WHERE txt LIKE '%базы данных%' \gx
```

Natija qanday bo'lishini taxmin qilish oson: 0 qator. Axir, LIKE operatori "базы" so'zini genitiv va akustiv shakllarda ("баз", "базы") tanimaydi. Quyidagiga binoan esa

```
test=# SELECT txt
FROM course_chapters
WHERE txt LIKE '%базу данных%' \gx
```

– ikkinchi bobdagi satr ko'rsatiladi, lekin I bobdan emas, chunki unda bu so'z boshqa kelishikda turibdi:

```
-[ RECORD 1 ]-------------------------------------
txt | Продолжаем знакомство с миром баз данных.
      Создадим нашу первую текстовую базу данных
```

Postgresda ILIKE operatori mavjud bo'lib, bu sizga hech bo'lmaganda katta va kichik harflar orasidagi farq haqida o'ylamaslik imkonini beradi. Albatta, yaratilishi san'at darajasida bo'lgan to'g'rilovchi (regular) iboralar (qidiruv qoliplari) ham mavjud, ammo ba'zida biz uchun o'ylaydigan vositaga ega bo'lishni xohlaymiz. Shuning uchun, boblar jadvaliga maxsus ma'lumotlar turi – tsvectorli yana bir ustun qo'shamiz:

```
test=# ALTER TABLE course_chapters
  ADD txtvector tsvector;
test=# UPDATE course_chapters
  SET txtvector = to_tsvector('russian',txt);
test=# SELECT txtvector
FROM course_chapters \gx

-[ RECORD 1 ]-------------------------------------
txtvector | 'баз':10 'глав':3 'дан':11 'знакомств':6
            'мир':9 'начина':4 'наш':5 'увлекательн':8
-[ RECORD 2 ]-------------------------------------
txtvector | 'баз':5,11 'дан':6,12 'знакомств':2
            'мир':4 'наш':8 'перв':9 'продолжа':1
            'создад':7 'текстов':10
-[ RECORD 3 ]-------------------------------------
txtvector | 'интриг':8 'мир':9 'начнет':2 'наш':3
            'полн':4 'приключен':5 'путешеств':6
            'сет':10
```

Biz ko'ramizki satrlarda:

1) so'zlar o'zgarmas qismlarga qisqartirildi (leksemma),
2) so'zning matndagi o'rnini ko'rsatadigan raqamlar paydo bo'ldi (ba'zi so'zlar ikki marta kiritilganligi ko'rinib turibdi),
3) predloglar qatorga kiritilmagan (shuningdek, qidiruv uchun ahamiyatsiz bo'lgan birikmalar va boshqa gap birliklarini – to'xtash so'zlarini o'z ichiga olmaydi).

Agar siz uning maydoniga bo'lim sarlavhalarini qo'shsangiz va shu bilan birga ularga matnning qolgan qismiga qaraganda kattaroq og'irlik bersangiz (bu `setweight` funksiyasi bilan amalga oshiriladi) qidiruv yanada yaxshi ishlaydi. Keling, jadvalni to'g'rilaymiz:

```
test=# UPDATE course_chapters
  SET txtvector =
      setweight(to_tsvector('russian',ch_title),'B')
      || ' ' ||
      setweight(to_tsvector('russian',txt),'D');

UPDATE 3
```

```
test=# SELECT txtvector
FROM course_chapters \gx

-[ RECORD 1 ]-------------------------------------
txtvector | 'баз':1B,12 'глав':5 'дан':2B,13
            'знакомств':8 'мир':11 'начина':6 'наш':7
            'увлекательн':10
-[ RECORD 2 ]-------------------------------------
txtvector | 'баз':7,13 'дан':8,14 'знакомств':4
            'мир':6 'наш':10 'перв':1B,11 'продолжа':3
            'создад':9 'текстов':12 'шаг':2B
-[ RECORD 3 ]-------------------------------------
txtvector | 'интриг':10 'локальн':1B 'мир':11
            'начнет':4 'наш':5 'полн':6 'приключен':7
            'путешеств':8 'сет':2B,12
```

Endi leksemalar nisbiy salmog'iga ega bo'ldi – B va D (mumkin bo'lgan to'rttadan – A, B, C, D). So'rovlarni yaratishda biz haqiqiy vaznni o'rnatamiz. Bu ularga qo'shimcha moslashuvchanlikni beradi.

Keling, to'liq qurollangan holda qidiruvga qaytaylik. `to_tsvector` funksiyasi `to_tsquery` funksiyasiga simmetrik bo'lib, u ramziy ifodani so'rovlarda qo'llaniladigan `tsquery` ma'lumotlar turiga aylantiradi.

```
test=# SELECT ch_title
FROM course_chapters
WHERE txtvector @@
      to_tsquery('russian','базы & данные');

  ch_title
-------------
 Базы данных
 Первые шаги
(2 rows)
```

Xuddi shu so'zlarning boshqa grammatik shakllari (`'база & данных'`) bilan qidiruv so'rovi bir xil natija berishiga ishonch hosil qilishingiz mumkin. Bu yerda biz `@@` taqqoslash ope-

ratoridan foydalandik, u to‘liq matnli qidiruv uchun LIKE – operatori odatdagi qidiruv uchun bajaradigan rolni bajaradi. @@ operatori tabiiy tildagi ifodalarda bo‘shliqlarni ishlatish uchun ruxsat bermaydi, shuning uchun so‘rovdagi so‘zlar mantiqiy AND operatori bilan birlashtiriladi.

russian argumenti MBBT tomonidan qo‘llaniladigan konfiguratsiyani ko‘rsatadi va qaysi lug‘atlarni ulash kerakligini va iborani leksemalarga ajratish uchun qaysi tahlilchidan foydalanish kerakligini belgilaydi.

Lug‘atlar o‘z nomiga qaramay, leksemalarni turli yo‘llar bilan o‘zgartirish imkonini beradi. Misol uchun, standart oddiy stemmer lug‘ati snowball so‘zning faqat o‘zgarmas qismini qoldiradi, shuning uchun qidiruv so‘rovdagi so‘zlarning oxirini e‘tiborsiz qoldiradi. Siz boshqalarini ham ulashingiz mumkin, masalan:

- morfologiyani aniqroq hisoblash uchun ispell, myspell yoki hunspell kabi “muntazam” lug‘atlar;
- sinonimlar lug‘ati;
- tezaurus;
- “ё” harfini “e”ga aylantirish uchun unaccent.

Belgilangan vaznlar tufayli yozuvlar reytingning kamayishi tartibida ko‘rsatiladi:

```
test=# SELECT ch_title,
     ts_rank_cd('{0.1, 0.0, 1.0, 0.0}', txtvector, q)
FROM course_chapters,
     to_tsquery('russian','базы & данных') q
WHERE txtvector @@ q
ORDER BY ts_rank_cd DESC;

  ch_title   | ts_rank_cd
-------------+------------
 Базы данных |    1.11818
 Первые шаги |       0.22
(2 rows)
```

0.1, 0.0, 1.0, 0.0 massivi ogʻirliklarni belgilaydi. Bu `ts_rank_cd` funksiyasining majburiy argumenti emas, sukut boʻyicha 0.1, 0.2, 0.4, 1.0 massivi D, C, B, A belgilarga toʻgʻri keladi. Soʻzning ogʻirligi topilgan qatorning ahamiyatiga taʻsir qiladi.

Yakuniy tajribada biz chiqishni oʻzgartiramiz. Biz html sahifasida qalin harf bilan topilgan soʻzlarni ajratib koʻrsatishni xohlaymiz deb taxmin qilamiz. `ts_headline` funktsiyasi soʻzni oʻrab oluvchi belgilar toʻplamini, shuningdek, qatordagi soʻzlarning minimal va maksimal sonini belgilaydi:

```
test=# SELECT ts_headline(
  'russian',
  txt,
  to_tsquery('russian', 'мир'),
  'StartSel=<b>, StopSel=</b>, MaxWords=50, MinWords=5'
)
FROM course_chapters
WHERE to_tsvector('russian', txt) @@
      to_tsquery('russian', 'мир');

-[ RECORD 1 ]----------------------------------------
ts_headline | знакомство с увлекательным <b>миром</b>
              баз данных
-[ RECORD 2 ]----------------------------------------
ts_headline | <b>миром</b> баз данных. Создадим нашу
-[ RECORD 3 ]----------------------------------------
ts_headline | путешествие в интригующий <b>мир</b>
              сетей
```

Umummatnli qidiruvni tezlashtirish uchun maxsus GiST, GIN va RUM indekslari qoʻllaniladi, ular maʻlumotlar bazasidagi oddiy indekslardan farq qiladi. Ammo ular, umummatnli qidiruv boʻyicha boshqa koʻplab foydali bilimlar singari, ushbu qisqa qoʻllanma doirasidan tashqarida qoladi.

Umummatnli qidiruv haqida batafsil maʻlumotni postgrespro.ru/doc/textsearch manzilidagi PostgreSQL hujjatlarida topishingiz mumkin.

JSON va JSONB bilan ishlash

SQLdan foydalanadigan relyatsion ma'lumotlar bazalari katta xavfsizlik chegarasi bilan yaratilgan: ularning iste'molchilarining birinchi tashvishi ma'lumotlarning yaxlitligi va xavfsizligi edi va ma'lumotlar hajmi zamonaviylari bilan taqqoslanmas edi. Ma'lumotlar bazasining yangi avlodi – NoSQL paydo bo'lganda, jamoa o'ylay boshladi: qat'iy izchillikni va sezilarli darajada soddalashtirilgan ma'lumotlar modelini qo'llab-quvvatlashdan bosh tortish (dastlab bu asosan kalit-qiymat juftliklari edi) qidiruvni sezilarli darajada tezlashtirishga imkon berdi. NoSQL ma'lumotlar bazalari misli ko'rilmagan hajmdagi ma'lumotlarni qayta ishlay olardi va parallel hisoblashdan to'liq foydalangan holda osonlikcha kengaytirilishi mumkin edi.

Birinchi zarba o'tib ketganda, ko'pchilikka haqiqiy hayot muammolari uchun oddiy tuzilma yetarli emasligi aniq bo'ldi. Murakkab kalitlar, so'ngra kalitlar guruhlari paydo bo'la boshladi. Relyatsion ma'lumotlar bazasini yaratuvchilari hayotdan orqada qolishni xohlamadilar va NoSQLga xos xususiyatlarni qo'shishni boshladilar.

Relyatsion MBBTlarda ma'lumotlar sxemasini o'zgartirish yuqori xarajatlar bilan bog'liq bo'lganligi sabab, foydali yangi ma'lumotlar turi – JSON paydo bo'ldi. U ierarxik tuzilmasi bo'yicha XMLga o'xshab, JavaScriptda dasturlash uchun mo'ljallangan (nomi ham shundan kelib chiqqan), shu jumladan, AJAX ilovalarini ishlab chiqish uchun ham. JSONning moslashuvchanligi dastur ishlab chiquvchilarga har safar ma'lumotlar bazasi sxemasini qayta ishlamasdan turib, oldindan aytib bo'lmaydigan tuzilmalarga ega ma'lumotlarni qo'shish imkonini berdi.

Aytaylik, endi siz shaxsiy ma'lumotlarni talabalar demo ma'lumotlar bazasiga kiritishingiz mumkin: biz so'rovnomani

ishga tushirdik va o'qituvchilardan so'radik. Anketaning barcha bandlarini to'ldirish shart emas, ba'zilari esa "boshqa" (другое) va "o'z ixtiyoringiz bilan ma'lumotlarni qo'shing" (добавьте данные на свое усмотрение) kabi javoblarga ruxsat beradi. An'anaviy yondashuv bilan, joriy tuzilishga mos kelmaydigan yangi ma'lumotlar ko'p sonli bo'sh maydonlarga ega bo'lgan ko'plab jadvallar yoki ustunlarni qo'shishni talab qiladi va tobora ko'proq yangi ma'lumotlarning paydo bo'lishi butun ma'lumotlarni doimiy ravishda qayta chizishga olib keladi.

Bu muammo `json` turi va keyinroq yaratilgan `jsonb` yordamida hal qilinadi, u ma'lumotlarni iqtisodiy ikkilik shaklda saqlaydi va `json`dan farqli o'laroq, indekslarni yaratish uchun moslashtiriladi, bu esa tartiblash bo'yicha qidiruvni tezlashtiradi.

Keling, JSON obyektlari bilan jadval yarataylik:

```
test=# CREATE TABLE student_details(
  de_id int,
  s_id int REFERENCES students(s_id),
  details json,
  CONSTRAINT pk_d PRIMARY KEY(s_id, de_id)
);
test=# INSERT INTO student_details
  (de_id, s_id, details) VALUES
(1, 1451,
'{ "достоинства": "отсутствуют",
   "недостатки":
   "неумеренное употребление мороженого",
   "статус": "отчислена"
}'),
(2, 1432,
'{ "хобби":
     { "гитарист":
         { "группа": "Постгрессоры",
           "гитары":["страт","телек"]
         }
     }
}'),
```

```
(3, 1556,
'{ "хобби": "косплей",
   "достоинства":
     { "мать-героиня":
         { "Вася": "м",
           "Семен": "м",
           "Люся": "ж",
           "Макар": "м",
           "Саша":"сведения отсутствуют"
         }
     }
}'
);
```

Keling, barcha ma'lumotlar joyida yoki yo'qligini tekshirib ko'ramiz. Qulaylik uchun, avval, `WHERE` bandidan foydalanib `student_details` va studentlar jadvallarini qo'shamiz, chunki yangi jadvalda talaba ismlari mavjud emas:

```
test=# SELECT s.name, sd.details
FROM student_details sd, students s
WHERE s.s_id = sd.s_id \gx
```

```
-[ RECORD 1 ]-------------------------------------
name    | Анна
details | { "достоинства": "отсутствуют",          +
        |    "недостатки":                         +
        |    "неумеренное употребление мороженого",+
        |    "статус": "отчислена"                 +
        | }
-[ RECORD 2 ]-------------------------------------
name    | Виктор
details | { "хобби":                               +
        |      { "гитарист":                       +
        |          { "группа": "Постгрессоры",     +
        |            "гитары":["страт","телек"]    +
        |          }                               +
        |      }                                   +
        | }
```

```
-[ RECORD 3 ]-------------------------------------
name    | Нина
details | { "хобби": "косплей",                    +
        |    "достоинства":                        +
        |      { "мать-героиня":                   +
        |          { "Вася": "м",                  +
        |            "Семен": "м",                 +
        |            "Люся": "ж",                  +
        |            "Макар": "м",                 +
        |            "Саша":"сведения отсутствуют" +
        |          }                               +
        |      }                                   +
        | }
```

Aytaylik, biz talabalarning xizmatlari haqida ma'lumotni o'z ichiga olgan yozuvlarga qiziqamiz. Keling, ->> operatoridan foydalanib, "достоинство" kalitining qiymatiga murojat qilamiz:

```
test=# SELECT s.name, sd.details
FROM  student_details sd, students s
WHERE s.s_id = sd.s_id
AND   sd.details ->> 'достоинства' IS NOT NULL \gx
-[ RECORD 1 ]-------------------------------------
name    | Анна
details | { "достоинства": "отсутствуют",          +
        |    "недостатки":                         +
        |    "неумеренное употребление мороженого",+
        |    "статус": "отчислена"                 +
        | }
-[ RECORD 2 ]-------------------------------------
name    | Нина
details | { "хобби": "косплей",                    +
        |    "достоинства":                        +
        |      { "мать-героиня":                   +
        |          { "Вася": "м",                  +
        |            "Семен": "м",                 +
        |            "Люся": "ж",                  +
        |            "Макар": "м",                 +
        |            "Саша":"сведения отсутствуют" +
        |          }                               +
        |      }                                   +
        | }
```

Ishonchimiz komil bo'ldiki, ikkita yozuv Anna va Ninaning xizmatlari bilan bog'liq, ammo bunday javob bizni qoniqtirishi dargumon: aslida Annaning xizmatlari "yo'q". Keling, so'rovni moslashtiramiz:

```
test=# SELECT s.name, sd.details
FROM  student_details sd, students s
WHERE s.s_id = sd.s_id
AND   sd.details ->> 'достоинства' IS NOT NULL
AND   sd.details ->> 'достоинства' != 'отсутствуют';
```

Ushbu so'rov ro'yxatda haqiqiy xarakatlari bor bo'lgan, Ninani qoldirishiga ishonch hosil qiling.

Ammo bu usul har doim ham ishlamaydi. Keling, musiqachi Vitya qanday gitara chalayotganini topishga harakat qilaylik:

```
test=# SELECT sd.de_id, s.name, sd.details
FROM  student_details sd, students s
WHERE s.s_id = sd.s_id
AND   sd.details ->> 'гитары' IS NOT NULL \gx
```

So'rov hech narsani qaytarmaydi. Gap shundaki, tegishli kalit-qiymat juftligi JSON ierarxiyasi ichida joylashgan, ya'ni yuqori darajadagi juftliklarga joylashtirilgan:

```
name    | Виктор
details | { "хобби":                                  +
        |       { "гитарист":                         +
        |           { "группа": "Постгрессоры",       +
        |             "гитары":["страт","телек"]      +
        |           }                                 +
        |       }                                     +
        | }
```

Gitaralarga o'tish uchun biz #> operatoridan foydalanamiz va "хобби"dan ierarxiya bo'ylab pastga tushamiz:

```
test=# SELECT sd.de_id, s.name,
       sd.details #> '{хобби,гитарист,гитары}'
FROM  student_details sd, students s
WHERE s.s_id = sd.s_id
AND   sd.details #> '{хобби,гитарист,гитары}'
      IS NOT NULL;
```

– va Viktor Fender muxlisi ekanligiga ishonch hosil qilamiz:

```
 de_id |  name  |     ?column?
-------+--------+-------------------
     2 | Виктор | ["страт","телек"]
```

json ma'lumotlar turi kichik ukasi jsonbga ega. “b” harfi ma'lumotlarni va uning tuzilishini saqlashning ikkilik (matn emas) usulini bildiradi, bu ko'p hollarda qidiruvni tezlashtiradi. So'nggi paytlarda jsonb jsonga qaraganda tez-tez ishlatiladi.

```
test=# ALTER TABLE student_details
ADD details_b jsonb;

test=# UPDATE student_details
SET details_b = to_jsonb(details);

test=# SELECT de_id, details_b
FROM student_details \gx
-[ RECORD 1 ]----------------------------------------
de_id     | 1
details_b | {"недостатки": "неумеренное употребление
            мороженого", "достоинства": "отсутствуют",
            "статус": "отчислена"}
-[ RECORD 2 ]----------------------------------------
de_id     | 2
details_b | {"хобби": {"гитарист": {"гитары":
            ["страт", "телек"], "группа":
            "Постгрессоры"}}}
-[ RECORD 3 ]----------------------------------------
de_id     | 3
details_b | {"хобби": "косплей", "достоинства":
            {"мать-героиня": {"Вася": "м", "Люся":
            "ж", "Саша": "сведения отсутствуют",
            "Макар": "м", "Семен": "м"}}}
```

Qayd etishning boshqa shakliga qo'shimcha ravishda, siz juftlikdagi qiymatlar tartibi o'zgarganini ham sezishingiz mumkin: Sasha, biz bilganimizdek uning haqida hech qanday ma'lumot yo'q. U endi ro'yxatda Makardan oldingi o'rinni egallaydi. Bu `jsonb`ning `json`ga nisbatan kamchiligi emas, balki axborotni saqlash xususiyati.

`json` bilan ishlashdan ko'ra `jsonb` bilan ishlash uchun operatorlar ko'proq. Eng foydalilaridan biri json uchun `#>`ga o'xshash `@>` obyektni ichiga kirishini tekshirish operatoridir.

Misol uchun, qahramonona qizi Lyusi tilga olingan yozuvni topamiz:

```
test=# SELECT s.name,
       jsonb_pretty(sd.details_b) json
FROM   student_details sd, students s
WHERE s.s_id = sd.s_id
AND    sd.details_b @>
       '{"достоинства":{"мать-героиня":{}}}' \gx

-[ RECORD 1 ]-------------------------------------
name | Нина
json | {                                          +
     |     "хобби": "косплей",                    +
     |     "достоинства": {                       +
     |         "мать-героиня": {                  +
     |             "Вася": "м",                   +
     |             "Люся": "ж",                   +
     |             "Саша": "сведения отсутствуют",+
     |             "Макар": "м",                  +
     |             "Семен": "м"                   +
     |         }                                  +
     |     }                                      +
     | }
```

Biz `jsonb` tipidagi chiqishni formatlaydigan `jsonb_pretty()` funksiyasidan foydalandik.

Yoki kalit-qiymat juftligini hosil qiluvchi `jsonb_each()` funksiyasidan foydalanishingiz mumkin:

```
test=# SELECT s.name,
       jsonb_each(sd.details_b)
FROM  student_details sd, students s
WHERE s.s_id = sd.s_id
AND   sd.details_b @>
      '{"достоинства":{"мать-героиня":{}}}' \gx

-[ RECORD 1 ]-------------------------------------
name       | Нина
jsonb_each | (хобби,"""косплей""")
-[ RECORD 2 ]-------------------------------------
name       | Нина
jsonb_each | (достоинства,"{""мать-героиня"":
             {""Вася"": ""м"", ""Люся"": ""ж"",
             ""Саша"": ""сведения отсутствуют"",
             ""Макар"": ""м"", ""Семен"": ""м""}}")
```

Aytgancha, Nina bolaning ismi o'rniga so'rovda bo'sh joy {} qoldirildi. Ushbu sintaksis real ilovalarni ishlab chiqish jarayoniga moslashuvchanlikni qo'shadi.

Lekin `jsonb`da asosiy narsa, ehtimol, `@>` operatorini, uning teskari `<@` va boshqalarni qo'llab-quvvatlaydigan indekslardir (GIN indeksi odatda eng samarali ishlaydi). `json` turi indekslarni qo'llab-quvvatlamaydi, shuning uchun og'ir yuklamali ilovalar uchun `jsonb`ni tanlash yaxshidir.

`json` va `jsonb` turlari va ular bilan ishlash funksiyalari haqida qo'shimcha ma'lumotni PostgreSQL hujjatlarining postgrespro.ru/doc/datatype-json va postgrespro.ru/doc/functions-json sahifalarida topishingiz mumkin.

SQL/JSON Path tilini o'z ichiga olgan SQL:2016 standarti chiqarilganda, Postgres Professional `jsonpath` turini qo'shadigan dasturni ishlab chiqdi. U PostgreSQL 12da paydo bo'ldi:

- `$.a.b.c` – buni PostgreSQL 11da siz `'a'->'b'->'c'` kabi yozishingiz kerak bo'ladi.

- $ – joriy kontekstni, ya'ni qayta ishlanayotgan JSON hujjat fragmentini ifodalaydi.
- @ – filtr ifodasidagi joriy element. $ ifodada mavjud bo'lgan yo'llar olinadi.
- * – metabelgi (wildcard). $ yoki @ lardagi ifodalarda ierarxiyani hisobga olgan holda yo'l bo'limining istalgan qiymatini olishni anglatadi.
- ** – $ yoki @ ifodaning bir qismi sifatida ierarxiyani hisobga olmagan holda har qanday yo'l segmenti qiymatini olishni anglatishi mumkin. Elementlarning joylashish darajasi noma'lum bo'lganda foydalidir.
- ? WHEREga o'xshash filtrni tashkil qilish imkonini beradi, masalan, `$.a.b.c ? (@.x > 10)`.

"косплей" ishqibozlarini qidirish uchun `jsonb_path_query()` funksiyasi bilan so'rov quyidagicha ko'rinishi mumkin:

```
test=# SELECT s_id, jsonb_path_query(
    details::jsonb,
    '$.хобби ? (@  == "косплей")'
)
FROM student_details;


 s_id | jsonb_path_query
------+------------------
 1556 | "косплей"
(1 row)
```

So'rov faqat "хобби" kaliti bilan boshlanadigan JSON tarmog'iga qaraydi va tegishli qiymat "косплей"ga teng yoki yo'qligini tekshiradi. Ammo agar biz "косплей"ni "гитарист" bilan almashtirsak, hech narsa ko'rsatilmaydi, chunki bizning jadvalimizda "гитарист" qiymat emas, balki o'rnatilgan yozuvning kalitidir.

So'rovda ikkita ierarxiya qo'llaniladi: biri qidiruvni cheklovchi $ ifodasi ichida ishlaydi, ikkinchisi esa @ ifodasi, ya'ni qidiruv vaqtida almashtiriladigan ifoda ichida ishlaydi. Bu sizga bir xil maqsadga turli yo'llar bilan erishish imkonini beradi. Masalan, bu so'rov

```
test=# SELECT s_id, jsonb_path_query(
  details::jsonb,
  '$.хобби.гитарист.группа?(@=="Постгрессоры")'
)
FROM student_details;
```

— va bunisi

```
test=# SELECT s_id, jsonb_path_query(
  details::jsonb,
  '$.хобби.гитарист?(@.группа=="Постгрессоры").группа'
)
FROM student_details;
```

– bir xil natijani beradi:

```
 s_id | jsonb_path_query
------+------------------
 1432 | "Постгрессоры"
(1 row)
```

Birinchi marta biz “хобби.гитарист.группа” filialidagi har bir yozuv uchun qidiruv doirasini o'rnatdik, agar siz JSONning o'ziga qarasangiz, bitta qiymatga – “Постгрессоры”ga mos keladi, shuning uchun tartiblash uchun hech narsa yo'q edi. Ikkinchi variantda “хобби.гитарист” tarmog'ida keladigan barcha filiallar bo'ylab yurib chiqish kerak edi, ammo filtr ifodasida biz “группа” tarmoq-yo'lini ko'rsatdik – aks holda yozuv topilmasdi. Ushbu sintaktik dizaynda biz JSON ichidagi ierarxiyani oldindan bilishimiz kerak. Ammo biz uni bilmasak nima qilish kerak?

Bunday holda, qo‘sh meta-belgi ** mos keladi. Juda ham foydali imkoniyat! Aytaylik, biz “строг” nima ekanligini unutib qo‘ydik – balandda uchadigan sharmi u, yoki gitara yoki yuqori ijtimoiy qatlam vakili, ammo bu so‘z bizning jadvalimizda bor yoki yo‘qligini aniqlashimiz kerak. JSON bilan oldingi amaliyotlardagi kabi yondashuv bilan murakkab so‘rov tuzishimiz kerak (agar siz jsonb turi bilan uni matnga aylantirmasdan ishlasangiz).

Endi bunday deyishimiz mumkin:

```
test=# SELECT s_id, jsonb_path_exists(
  details::jsonb,
  '$.** ? (@ == "строг")'
)
FROM student_details;
```

```
 s_id | jsonb_path_exists
------+-------------------
 1451 | f
 1432 | t
 1556 | f
 1451 | f
(4 rows)
```

SQL/JSON Path imkoniyatlarini nafaqat hujjatlarda (postgrespro.ru/doc/datatype-json#DATATYPE-JSONPATH), balki (habr.com/ru/company/postgrespro/blog/448612/) maqolasida ham topish mumkin.

SQL:2016 standarti JSON bilan ishlash uchun boshqa ko‘plab qulay funksiya va ifodalarni taqdim etadi. Ularning ilovalari asta-sekin PostgreSQLga qo‘shilmoqda. Biz barcha yangi funksiyalar va operatorlarni sanab o‘tmaymiz, ular haqida hujjatlarda o‘qishingiz mumkin: postgrespro.ru/docs/postgresql/16/functions-json.

§Keling, bizning talabalar guruhining hikoyasini davom ettiramiz – Anna, Viktor va Ninalar qahramonlarimiz. Ularning oldiga media-sotsiologlar kelib, yoshlar qanday serial tomosha qilayotganini bilishdi. Natijalarni jadvalda qayd etishdi:

```
test=# CREATE TABLE tv_series(
  se_id int,
  s_id int REFERENCES students(s_id),
  se_details jsonb,
  CONSTRAINT pk_se PRIMARY KEY(s_id, se_id)
);

test=# INSERT INTO tv_series (se_id, s_id, se_details)
VALUES
(1, 1451,
'{"название": ["Игра престолов",
               "Слово пацана"]}'
),
(2, 1432,
'{"название": "17 мгновений весны"}'
),
(3, 1556,
'{"название": ["Twin peaks",
               "Слово пацана"]}'
);
```

Dasturchilar natijalarni qayta ishlaydilar. Endi ular quyidagi kuchli vositalarga ega: agregatsiya funktsiyalari, tekshirish predikatlari, `json(b)` elementlarni manipulyatsiya qilish vositalari.

Massiv elementlarini `jsonb_array_elements` funktsiyasi yordamida alohida yozuvlarga bo'lish (`unnest`) kerak, ammo bu yerda muammo bor:

```
text=# SELECT jsonb_array_elements(
  se_details->'название'
)
FROM tv_series;

ERROR:  cannot extract elements from a scalar
```

Albatta, ma'lumotlar so'rovlardan ma'lumotlar bazasiga noto'g'ri uzatilgan: talabaning 1432 raqami massiv emas. Buni IS [NOT] JSON ARRAY predikati yordamida tekshiramiz:

```
text=# SELECT se_id, se_details
FROM tv_series
WHERE se_details->'название' IS NOT JSON ARRAY \gx
```

```
-[ RECORD 1 ]---------------------------------
se_id      | 2
se_details | {"название": "17 мгновений весны"}
```

Keling, jsonbning ichki qismlariga kira oladigan jsonb_set funktsiyasidan foydalanib turni to'g'rilaymiz:

```
text=# UPDATE tv_series
SET se_details = jsonb_set(
  se_details, '{название}', '["17 мгновений весны"]'
)
WHERE se_id = 2;
```

Endi:

```
test=# SELECT DISTINCT jsonb_array_elements(
  se_details->'название'
)
FROM tv_series;
```

```
 jsonb_array_elements
----------------------
 "Twin peaks"
 "17 мгновений весны"
 "Слово пацана"
 "Игра престолов"
(4 rows)
```

Ro'yxat tayyor. E'tibor bering, IS JSON ARRAY predikati IS JSON SCALAR, IS JSON OBJECT va umumiyroq IS JSONdan iborat o'xshashlar guruhining bir qismidir. Sizga eslatib o'tamizki, siz ular haqida hujjatlarda o'qishingiz mumkin: postgrespro.ru/docs/postgresql/16/functions-json.

Tashqi tizimlar bilan integratsiya

Ilovalar yolg‘iz emas, balki o‘z turlari orasida yashaydi va ko‘pincha bir-biri bilan muloqot qiladi. Bunday aloqa ilovalarning o‘zlari yordamida amalga oshirilishi mumkin, masalan, veb-xizmatlar yoki fayl almashinuvi yordamida yoki siz MBBT vositalaridan foydalanishingiz mumkin.

PostgreSQL maxsus tashqi ma‘lumotlarni o‘rash (foreign data wrapper) mexanizmi orqali tashqi ma‘lumot manbalari bilan SQLda ishlash uchun ISO/IEC 9075-9 (SQL/MED, Tashqi ma‘lumotlarni boshqarish) standartini qo‘llab-quvvatlaydi.

Mexanizmning g‘oyasi shundan iboratki, tashqi (uchinchi tomon) ma‘lumotlariga oddiy jadvallar kabi kirish mumkin. Buning uchun birinchi navbatda uchinchi tomon jadvallari (xorijiy jadvallar) yaratiladi, ular o‘zlari ma‘lumotlarni o‘z ichiga olmaydi, lekin barcha chaqirishlarni tashqi manbaga yo‘naltiradi. Ushbu yondashuv ilovalarni ishlab chiqishni soddalashtiradi, chunki u ma‘lum bir tashqi manba bilan ishlashning o‘ziga xos xususiyatlarini bilishni talab qilmaydi.

Uchinchi tomon jadvallarini yaratish jarayoni bir necha ketma-ket bosqichlardan iborat.

1. `CREATE FOREIGN DATA WRAPPER` buyrug‘i yordamida kutubxonani ma‘lum bir ma‘lumot manbai bilan ishlashga ulaymiz.

2. `CREATE SERVER` buyrug‘i yordamida tashqi ma‘lumotlar manbai joylashgan serverni aniqlaymiz. Buning uchun buyruq odatda server nomi, port raqami va ma‘lumotlar bazasi nomi kabi parametrlarni ko‘rsatadi.

3. Turli PostgreSQL foydalanuvchilari bir xil tashqi ma'lumotlar manbasiga turli hisoblar yordamida ulanishlari mumkin, shuning uchun nom xaritasini belgilash uchun `CREATE USER MAPPING` buyrug'idan foydalaning.

4. Masofaviy serverning kerakli jadvallari va ko'rinishlari uchun biz `CREATE FOREIGN TABLE` buyrug'i bilan uchinchi tomon jadvallarini yaratamiz va `IMPORT FOREIGN SCHEMA` buyrug'i belgilangan sxemadan jadvallarning barcha yoki bir qismi tavsiflarini import qilish imkonini beradi.

Biz PostgreSQLning eng mashhur MBBTlar bilan integratsiyasini ko'rib chiqamiz: Oracle, MySQL, SQL Server va PostgreSQL. Lekin birinchi navbatda ma'lumotlar bazalari bilan ishlash uchun tegishli kutubxonalarni o'rnatishingiz kerak.

Kengaytmalarni o'rnatish

PostgreSQL distributiviga ikkita uchinchi tomon ma'lumotlar paketini o'z ichiga oladi: `postgres_fdw` va `file_fdw`. Birinchisi masofaviy PostgreSQL ma'lumotlar bazalari bilan, ikkinchisi – serverdagi fayllar bilan ishlash uchun mo'ljallangan. Bundan tashqari, jamoa ko'plab umumiy ma'lumotlar bazasi tizimlariga kirish uchun kutubxonalarni ishlab chiqdi va ularga xizmat ko'rsatdi. Ularning ro'yxatini pgxn.org/tag/fdw saytida ko'rish mumkin.

Oracle, MySQL va SQL Server uchun uchinchi tomon ma'lumotlar o'ramlari kengaytmalar sifatida mavjud:

- Oracle — github.com/laurenz/oracle_fdw;
- MySQL — github.com/EnterpriseDB/mysql_fdw;
- SQL Server — github.com/tds-fdw/tds_fdw.

Ushbu saytlardagi ko'rsatmalarga rioya qiling shunda yig'ish va o'rnatish muammosiz bo'ladi. Agar hamma narsa to'g'ri bajarilgan bo'lsa, tegishli uchinchi tomon ma'lumotlar paketlari mavjud kengaytmalar ro'yxatida paydo bo'ladi. Masalan, `oracle_fdw` uchun:

```
test=# SELECT name, default_version
FROM pg_available_extensions
WHERE name = 'oracle_fdw' \gx
```

```
-[ RECORD 1 ]---+-----------
name            | oracle_fdw
default_version | 1.2
```

Oracle

Avvaliga, kengaytmani o'rnating, bu esa o'z navbatida uchinchi tomon ma'lumotlari uchun o'ramni yaratadi:

```
test=# CREATE EXTENSION oracle_fdw;
```

```
CREATE EXTENSION
```

Tegishli o'ram yaratilganligini tekshiramiz:

```
test=# \dew
```

```
List of foreign-data wrappers
-[ RECORD 1 ]-------------------
Name      | oracle_fdw
Owner     | postgres
Handler   | oracle_fdw_handler
Validator | oracle_fdw_validator
```

Keyingi qadam uchinchi tomon ma'lumotlar serverini yaratishdir. `OPTIONS` bandi `dbserver` parametrini belgilaydi, u Oraclega ulanish uchun xos ma'lumotlarni aniqlashni so'raydi: server nomi, port raqami va misol nomi.

```
test=# CREATE SERVER oracle_srv
  FOREIGN DATA WRAPPER oracle_fdw
  OPTIONS (dbserver '//localhost:1521/orcl');

CREATE SERVER
```

PostgreSQL foydalanuvchisi `postgres` Oracle ga `scott` sifatida ulanadi.

```
test=# CREATE USER MAPPING FOR postgres
  SERVER oracle_srv
  OPTIONS (user 'scott', password 'tiger');

CREATE USER MAPPING
```

Biz uchinchi tomon jadvallarini alohida sxemaga import qilamiz. Keling, uni yarataylik:

```
test=# CREATE SCHEMA oracle_hr;

CREATE SCHEMA
```

Masofaviy jadvallar tavsiflarini import qilish. Keling, ikkita mashhur `dept` va `emp` jadvallari bilan cheklanamiz:

```
test=# IMPORT FOREIGN SCHEMA "SCOTT"
  LIMIT TO (dept, emp)
  FROM SERVER oracle_srv
  INTO oracle_hr;

IMPORT FOREIGN SCHEMA
```

E'tibor bering, Oracle ma'lumotlar lug'atida obyekt nomlari katta harflarda, PostgreSQL tizim katalogida esa kichik harflarda saqlanadi. Shuning uchun, PostgreSQL-da tashqi ma'lumotlar bilan ishlaganda, kichik harflarga o'tkazmaslik uchun Oracle sxemasi nomini katta harflar va qo'shtirnoq ichida yozing.

Keling, uchinchi tomon jadvallari ro'yxatini ko'rib chiqaylik:

```
test=# \det oracle_hr.*
```

```
     List of foreign tables
  Schema   | Table |   Server
-----------+-------+------------
 oracle_hr | dept  | oracle_srv
 oracle_hr | emp   | oracle_srv
(2 rows)
```

Endi masofaviy ma'lumotlarni olish uchun biz uchinchi tomon jadvallariga so'rovlarni bajaramiz:

```
test=# SELECT * FROM oracle_hr.emp LIMIT 1 \gx
```

```
-[ RECORD 1 ]-------------------
empno    | 7369
ename    | SMITH
job      | CLERK
mgr      | 7902
hiredate | 1980-12-17
sal      | 800.00
comm     |
deptno   | 20
```

Ma'lumotlarni faqatgina o'qish emas, balki o'zgartirish ham mumkin:

```
test=# INSERT INTO oracle_hr.dept(deptno, dname, loc)
  VALUES (50, 'EDUCATION', 'MOSCOW');
```

```
INSERT 0 1
```

```
test=# SELECT * FROM oracle_hr.dept;
```

```
 deptno |   dname    |   loc
--------+------------+----------
     10 | ACCOUNTING | NEW YORK
     20 | RESEARCH   | DALLAS
     30 | SALES      | CHICAGO
     40 | OPERATIONS | BOSTON
     50 | EDUCATION  | MOSCOW
(5 rows)
```

Biz kengaytma va u bilan birga uchinchi tomon ma'lumotlari uchun o'ramni yaratamiz:

```
test=# CREATE EXTENSION mysql_fdw;
CREATE EXTENSION
```

Ulanadigan nusxani tavsiflovchisi uchinchi tomon serveri `host` va `port` parametrlari bilan aniqlanadi:

```
test=# CREATE SERVER mysql_srv
    FOREIGN DATA WRAPPER mysql_fdw
    OPTIONS (host 'localhost', port '3306');
CREATE SERVER
```

Biz MySQL superuseri bo'lib ulanamiz:

```
test=# CREATE USER MAPPING FOR postgres
  SERVER mysql_srv
  OPTIONS (username 'root', password 'p@ssw0rd');
CREATE USER MAPPING
```

Qobiq `IMPORT FOREIGN SCHEMA` buyrug'ini qo'llab-quvvatlaydi, ammo siz tashqi jadvalni qo'lda yaratishingiz mumkin:

```
test=# CREATE FOREIGN TABLE employees (
  emp_no      int,
  birth_date  date,
  first_name  varchar(14),
  last_name   varchar(16),
  gender      varchar(1),
  hire_date   date)
SERVER mysql_srv
  OPTIONS (dbname 'employees',
           table_name 'employees');
CREATE FOREIGN TABLE
```

Tekshiramiz:

```
test=# SELECT * FROM employees LIMIT 1 \gx

-[ RECORD 1 ]----------
emp_no     | 10001
birth_date | 1953-09-02
first_name | Georgi
last_name  | Facello
gender     | M
hire_date  | 1986-06-26
```

Oracle singari, `mysql_fdw` o'rami nafaqat o'qish, balki ma'lumotlarni o'zgartirish imkonini beradi.

SQL Server

Kengaytmani va u bilan birga uchinchi tomon ma'lumotlari uchun o'ramni yaratamiz:

```
test=# CREATE EXTENSION tds_fdw;

CREATE EXTENSION
```

Uchinchi tomon serverni yaratamiz:

```
test=# CREATE SERVER sqlserver_srv
  FOREIGN DATA WRAPPER tds_fdw
  OPTIONS (servername 'localhost', port '1433',
           database 'AdventureWorks');

CREATE SERVER
```

Taqdim etilgan ma'lumotlar o'zgarmaydi: siz server nomini, port raqamini, ma'lumotlar bazasini ko'rsatishingiz kerak. Ammo `OPTIONS` bandidagi parametrlarning soni va nomlari biz `oracle_fdw` va `mysql_fdw` uchun ko'rganimizdan farq qiladi.

Biz SQL Serverning superuser hisobi ostida ulanamiz:

```
test=# CREATE USER MAPPING FOR postgres
    SERVER sqlserver_srv
    OPTIONS (username 'sa', password 'p@ssw0rd');

CREATE USER MAPPING
```

Keling, uchinchi tomon jadvallari uchun alohida sxema yaratay-lik:

```
test=# CREATE SCHEMA sqlserver_hr;

CREATE SCHEMA
```

Biz HumanResources sxemasini butunlayicha yaratilgan Post-greSQL sxemasiga import qilamiz:

```
test=# IMPORT FOREIGN SCHEMA HumanResources
    FROM SERVER sqlserver_srv
    INTO sqlserver_hr;

IMPORT FOREIGN SCHEMA
```

Import qilingan jadvallar ro'yxatini \det buyrug'i bilan tek-shirish mumkin yoki tizim katalogida quyidagi so'rov bilan topish mumkin:

```
test=# SELECT ft.ftrelid::regclass AS "Table"
FROM pg_foreign_table ft;

                  Table
----------------------------------------
 sqlserver_hr.Department
 sqlserver_hr.Employee
 sqlserver_hr.EmployeeDepartmentHistory
 sqlserver_hr.EmployeePayHistory
 sqlserver_hr.JobCandidate
 sqlserver_hr.Shift
(6 rows)
```

Obyekt nomlari katta-kichik harflarga sezgir, shuning uchun ular PostgreSQLda qo'sh tirnoq ichida havola qilinishi kerak:

```
test=# SELECT "DepartmentID", "Name", "GroupName"
FROM sqlserver_hr."Department"
LIMIT 4;

 DepartmentID |    Name     |        GroupName
--------------+-------------+------------------------
            1 | Engineering | Research and Development
            2 | Tool Design | Research and Development
            3 | Sales       | Sales and Marketing
            4 | Marketing   | Sales and Marketing
(4 rows)
```

Hozirda `tds_fdw` faqat o'qishni qo'llab-quvvatlaydi, ma'lumotlarni o'zgartirmaydi.

PostgreSQL

Kengaytma va o'ramni yaratamiz:

```
test=# CREATE EXTENSION postgres_fdw;

CREATE EXTENSION
```

Biz xuddi shu klasterning boshqa ma'lumotlar bazasiga ulanamiz, shuning uchun uchinchi tomon serverini yaratishda faqat `dbname` parametrini ko'rsatish kifoya qiladi va `host`, `port` va boshqa parametrlarni o'tkazib yuborish mumkin:

```
test=# CREATE SERVER postgres_srv
  FOREIGN DATA WRAPPER postgres_fdw
  OPTIONS (dbname 'demo');

CREATE SERVER
```

Xuddi shu ma'lumotlar bazasi klasterining foydalanuvchilarini moslashtirganda parolni ko'rsatish shart emas:

```
test=# CREATE USER MAPPING FOR postgres
  SERVER postgres_srv
  OPTIONS (user 'postgres');

CREATE USER MAPPING
```

bookings sxemasiga tegishli barcha jadvallar va ko'rinishlarni import qilamiz:

```
test=# IMPORT FOREIGN SCHEMA bookings
  FROM SERVER postgres_srv
  INTO public;

IMPORT FOREIGN SCHEMA
```

Tekshiramiz:

```
test=# SELECT * FROM bookings LIMIT 3;

 book_ref |       book_date        | total_amount
----------+------------------------+--------------
 000004   | 2015-10-12 14:40:00+03 |     55800.00
 00000F   | 2016-09-02 02:12:00+03 |    265700.00
 000010   | 2016-03-08 18:45:00+03 |     50900.00
 000012   | 2017-07-14 09:02:00+03 |     37900.00
 000026   | 2016-08-30 11:08:00+03 |     95600.00
(5 rows)
```

postgres_fdw haqida ko'proq hujjatlarda o'qishingiz mumkin: postgrespro.ru/doc/postgres-fdw.

Uchinchi tomon ma'lumotlarini o'rash mexanizmi ham qiziq, chunki u hamjamiyat tomonidan PostgreSQLga o'rnatilgan shardingni yaratish uchun asos sifatida ko'rib chiqiladi. Sharding qismlarga ajratishga o'xshaydi: ikkala holatda ham jadval qandaydir xarakteristikaga ko'ra bir-biridan alohida saqlanadigan bir nechta qismlarga bo'linadi. Farqi shundaki,

bo‘limlar bir xil serverda, shardlar esa boshqalarida joylashgan. PostgreSQLda bo‘linish ancha vaqtdan beri mavjud. 10-talqindan boshlab ushbu mexanizm faol rivojlanmoqda: deklarativ sintaksis qo‘shildi, bo‘limlarni dinamik ravishda chiqarib tashlash, parallel ishlov berish va boshqa yaxshilanishlar amalga oshirildi. Tashqi jadvallar bo‘limlar sifatida ham ishlatilishi mumkin, shuning uchun bo‘linish parchalanishga aylanadi.

Sharding haqiqatan ham ishlatilishi uchun bu yo‘lda – hali ko‘p ish qilinishi kerak:

- izchillik kafolatlanmaydi: tashqi serverlar bilan ishlash bitta taqsimlangan tranzaksiyada emas, balki alohida lokal tranzaksiyalarda amalga oshiriladi;
- nosozliklarga chidamliligini oshirish uchun bir xil ma‘lumotlarni bir nechta serverlarda takrorlash imkoniyati yo‘q;
- shardlar va mos keladigan tashqi jadvallar bo‘yicha jadvallar yaratish uchun barcha kerakli harakatlar hali ham qo‘lda bajarilishi kerak.

Ro‘yxatdagi vazifalar bizning Shardman vositamizda (postgrespro.ru/products/shardman) hal qilingan.

PostgreSQL ma‘lumotlar bazalari bilan ishlash uchun distributivda yana bir kengaytma mavjud – `dblink`. U ulanishlarni aniq boshqarish (ulanish, uzish), so‘rovlarni bajarish va asinxron natijalarni olish imkonini beradi: postgrespro.ru/doc/dblink.

X Ta‘lim va sertifikatlash

Hujjatlar

PostgreSQL bilan jiddiy ishlash uchun hujjatlarni o'qish zarur. Bu faqat MBBTning barcha imkoniyatlari tavsifi emas, balki har doim qo'lingizda bo'lishi kerak bo'lgan to'liq ma'lumotnoma qo'llanmasidir. Hujjatlarni o'qib, siz birlamchi manbadan qisqa va aniq ma‘lumotlarni olasiz — ular to‘g‘ridan-to‘g‘ri ishlab chiquvchilar tomonidan yozilgan va har doim dolzarbligini saqlab turadi.

Bizning Postgres Professional kompaniyamizda PostgreSQLning barcha hujjatlari to‘plami, jumladan, eng so‘nggi talqini ham rus tiliga tarjima qilingan — ular postgrespro.ru/docs saytida mavjud.

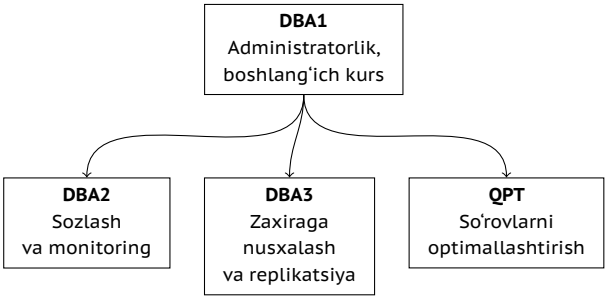
Bizning tarjima uchun tuzgan lug‘atimiz postgrespro.ru/education/glossary manzilida e'lon qilingan. Ingliz tilidagi hujjatlarni to‘g‘ri tarjima qilish va rus tilidagi materiallar uchun hamma uchun tushunarli yagona terminologiyani qo‘llash uchun uni ishlatishni tavsiya qilamiz.

Ingliz tilidagi asl hujjatlarni afzal ko‘rganlar uni bizning saytimizda, shuningdek, postgresql.org/docs manzilida topishlari mumkin.

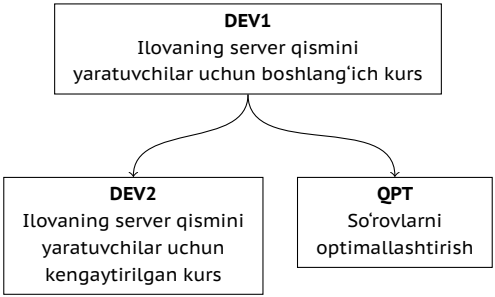
O‘quv kurslari

Biz PostgreSQL bilan ishlashni boshlayotgan yoki malakasini oshirayotganlar uchun o‘quv kurslarini ishlab chiqmoqdamiz.

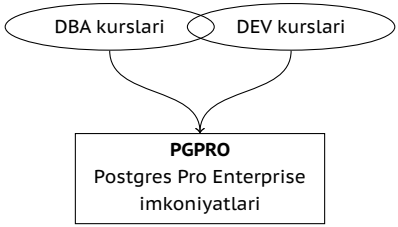
Ma‘lumotlar bazasi administratorlari uchun kurslar:



Va amaliyotchi dasturchilar uchun:



Barcha ma'lumotlar bazasi administratorlik kurslarini yoki ilova dasturlash kurslarini o'zlashtirganlar uchun biz yana bir kursni taklif etamiz, bu bizning asosiy mahsulotimizga bag'ishlangan:



PostgreSQL va Postgres Pro hujjatlari ish uchun zarur bo'lgan barcha ma'lumotlarni o'z ichiga oladi, biroq ular turli bo'limlarga tarqalgan va ko'p marotaba diqqat bilan o'qishni talab qiladi.

Kurslar hujjatlarni o'rnini bosmaydi, balki ularni to'ldiradi. O'quv modullari izchil va bog'langan tarzda mazmunni ochib beradi, muxim va amaliy jihatdan foydali ma'lumotlarni ajratib ko'rsatadi. O'quv kurslaridan o'tish kerakli bilimlarni kengaytiradi, avvalgi olingan uzluksiz ma'lumotlarni tizimlashtiradi va hujjatlar bo'yicha yaxshiroq yo'nalishga yordam beradi hamda kerakli tafsilotlarni tezda aniqlashga imkon beradi.

Har bir kurs mavzusi nazariy qism va amaliyotdan iborat. Nazariy qism ko'p hollarda nafaqat taqdimot, balki "jonli" tizimda ishlashning namoyishini ham o'z ichiga oladi. Kurs tinglovchilari har bir slayd batafsil sharhlar bilan yoritilgan taqdimotlar, namoyish skriptlarining ishlash natijalari, amaliy masalalarning yechimlari va ba'zi hollarda qo'shimcha ma'lumotnoma materiallarini oladilar.

Qayerda va qanday ta'lim olish mumkin

Mustaqil o'qish va notijorat foydalanish uchun barcha kurs materiallari, shu jumladan videoyozuvlar, hamma uchun bizning saytimizda mavjud. Ularni postgrespro.ru/education/courses manzilida topishingiz mumkin.

Shuningdek, siz yuqorida sanab o'tilgan kurslarni tajribali o'qituvchi rahbarligida maxsus o'quv markazlarida o'qishingiz mumkin. Kursni tugatgandan so'ng, tinglovchi sertifikati beriladi. Biz tomonimizdan tasdiqlangan o'quv markazlarining ro'yxati: postgrespro.ru/education/where.

DBA1. PostgreSQL administratorligi bo'yicha boshlang'ich kurs

Davomiylik: 3 kun

Talab etilgan bilimlar:

Ma'lumotlar bazalari va SQL haqida minimum tushunchalar.
Unix bilan tanish bo'lish.

Qanday ko'nikmalar olinadi:

PostgreSQL arxitekturasi haqida umumiy ma'lumot.
O'rnatish, boshlang'ich sozlashlar, serverni boshqarish.
Ma'lumotlarni mantiqiy va fizik darajada tashkil etish.
Administratorlik boshlang'ich vazifalari.
Foydalanuvchilarni va kirish imkoniyatlarini boshqarish.
Zaxira uchun nusxalash, tiklash va replikatsiya haqida tushuncha.

Mavzular:

Asosiy vositalar

1. Serverni o'rnatish va boshqarish
2. Psqldan foydalanish
3. Konfiguratsiyalash

Arxitektura

4. PostgreSQL umumiy tuzilishi
5. Izolyatsiya va ko'ptalqinlilik
6. Bufer kesh va jurnal

Ma'lumotlarni tashkillashtirish

7. Ma'lumotlar bazasi va sxemalar
8. Tizim katalogi
9. Jadval mayonlari
10. Quyi qism

Administratorlik vazifalari

11. Monitoring
12. Boshchilik qilish

Kirishlarni boshqarish

13. Rollar va atributlar
14. Imtiyozlar
15. Qatorlarni himoya qilish siyosati
16. Ulanish va autentifikatsiya

Zaxira uchun nusxalash

17. Umumiy sharh

Replikatsiya

18. Umumiy sharh

O'quv kursi materiallari mustaqil o'rganish uchun postgrespro.ru/education/courses/DBA1 manzilida mavjud.

160
x

DBA2. PostgreSQLni sozlash va monitoring qilish

Davomiylik: 4 kun

Talab etilgan bilimlar:

SQL tili asoslari.
Unix OTda ishlay olish.
PostgreSQL bilan DBA1 kursi hajmi darajasida tanish bo'lish.

Qanday ko'nikmalar olinadi:

Serverning ichki tuzilishini tushunish asosida turli konfiguratsiya parametrlarini sozlash.
Serverni monitoring qilish va parametrlarni iterativ sozlash uchun olingan ma'lumotlardan foydalanish.
Mahalliylashtirish bilan bog'liq sozlamalar.
Kengaytmalarni boshqarish va serverni yangilash jarayoni bilan tanishuv.

Mavzular:

Ko'ptalqinlilik

1. Izolyatsiya
2. Sahifalar va qatorlar talqinlari
3. Ma'lumotlarning sur'atlari
4. HOT-yangilanishlar
5. Tozalash
6. Avtotozalash
7. Muzlatish

Jurnal yuritish

8. Bufer kesh
9. Oldindan yozish jurnali
10. Nazorat nuqtasi
11. Jurnalni sozlash

Bloklashlar

12. Obyektlarni bloklash
13. Qatorlarni bloklash
14. Tezkor xotiradagi bloklashlar

Administratorlik vazifalari

15. Kengaytmalarni boshqarish
16. Mahalliylashtirish
17. Serverni yangilash

O'quv kursi materiallari mustaqil o'rganish uchun postgrespro.ru/education/courses/DBA2 manzilida mavjud.

DBA3. PostgreSQLda zaxira uchun nusxalash va replikatsiya

Davomiylik: 2 kun

Talab etilgan bilimlar:

SQL tili asoslari.
Unix OTdan foydalana olish.
PostgreSQL bilan DBA1 kurs hajmida tanish bo'lish.

Qanday ko'nikmalari olinadi:

Zaxira uchun nusxalashni bajarish.
Serverni mantiqiy va fizik replikatsiyalar uchun sozlash.
Replikatsiyalardan foydalanish senariylari bilan tanishish.
Klasterlarni qurish usullari yuzasidan tushuncha.

Mavzular:

Zaxira uchun nusxalash

1. Mantiqiy nusxalash
2. Asosiy zaxira uchun nusxalash
3. Oldindan yozish jurnalini nusxalash

Replikatsiya

4. Fizik replikatsiya
5. Replikatsiyaga ulanish
6. Mantiqiy replikatsiya
7. Foydalanish senariylari

Klasterli texnologiyalar

8. Umumiy sharh

O'quv kursi materiallari mustaqil o'rganish uchun postgrespro.ru/education/courses/DBA3 manzilida mavjud.

DEV1. Ilova dasturlarning server qismini ishlab chiqish bo'yicha boshlang'ich kurs

Davomiylik: 4 kun

Talab etilgan bilimlar:

SQL tili asoslari.
Protsedura dasturlash tillaridan biri bilan ishlash yuzasidan tajriba.
Unix ishlashi haqida minimal tushunchalarga ega bo'lish.

Qanday ko'nikmalar olinadi:

PostgreSQL arxitekturasi haqida umumiy ma'lumotlar.
MBning asosiy obyektlarini ishlatish.
SQL va PL/pgSQL tillarida server tomonda dasturlash.
Yozuvlar va massivlarni ham qo'llagan holda asosiy turlardan foydalanish.
Mijoz qism bilan o'zaro aloqani tashkillashtirish.

Mavzular:

Asosiy vositalar

1. O'rnatish va boshqaruv; psql

Arxitektura

2. PostgreSQLning umumiy tuzilishi
3. Izolyatsiya va ko'ptalqinlik
4. Bufer kesh va jurnal

Ma'lumotlarni tashkillashtirish

5. Mantiqiy tuzilma
6. Fizik tuzilma

"Kitoblar do'koni" Ilovasi

7. Ilovaning ma'lumotlar sxemasi

SQL

8. Funksiyalar
9. Protseduralar
10. Tashkiliy turlar

PL/pgSQL

11. Til umumiy ko'rinishi va kostruksiyalari
12. So'rovlarning bajarilishi

13. Kursorlar
14. Dinamik buyruqlar
15. Massivlar
16. Xatoliklarni qayta ishlash
17. Triggerlar
18. Xatoliklarni tutish va nosozliklarni tuzatish

Kirishlarni chegaralash

19. Kirishlarni chegaralashni umumiy sharhi

Zaxira uchun nusxalash

20. Mantiqiy zaxiralash

O'quv kursi materiallari mustaqil o'rganish uchun postgrespro.ru/education/courses/DEV1 manzilida mavjud.

DEV2. Ilova dasturlarning server qismini ishlab chiqish bo'yicha kengaytirilgan kurs

Davomiylik: 4 kun

Talab etilgan bilimlar:

PostgreSQL arxitekturasi haqida umumiy tushunchalar.
SQL va PL/pgSQL bo'yicha mustahkam bilimlar.
Unixda ishlash haqida minimum tushunchalar.

Qanday ko'nikmalar olinadi:

Serverning ichki tuzilishini tushunish.
Ilova mantig'ini amalga oshirish uchun PostgreSQL tomonidan taqdim etilgan imkoniyatlardan to'liq foydalanish.

Maxsus vazifalarni hal qilish uchun MBBT imkoniyatlarini kengaytirish. 165 x

Mavzular:

Arxitektura

1. Izolyatsiya
2. Ichki qurilma
3. Tozalash
4. Jurnal yuritish
5. Bloklashlar

“Kitoblar do‘koni”

6. 2.0 Ilova

Kengayuvchanlik

7. Ulanishlar puli
8. Katta qiymatlar uchun turlar
9. Foydalanuvchi ma‘lumot turlari
10. Operatorlar sinflari
11. Zaiftuzilmali ma‘lumotlar
12. Orqa tomonda ishlaydigan jarayonlar
13. Asinxron qayta ishlash
14. Kengaytmalarni yaratish
15. Dasturlash tillari
16. Agregat va oyna funksiyalari
17. Umummatn qidiruvi
18. Fizik replikatsiya
19. Mantiqiy replikatsiya
20. Tashqi ma‘lumotlar

O‘quv kursi materiallari mustaqil o‘rganish uchun postgrespro.ru/education/courses/DEV2 manzilida mavjud.

166
x

QPT. PostgreSQL so'rovlarini maqbullashtirish

Davomiylik: 2 kun

Talab etilgan bilimlar:

Unix OT bilan tanish bo'lish.
SQL bo'yicha mustahkam bilimlar.
PL/pgSQL tilini bilish foydali bo'ladi, ammo zarur emas.
PostgreSQL bilan DBA1 (administratorlar uchun) yoki DEV1 (dastur yaratuvchilar uchun) kurs hajmida tanish bo'lish.

Qanday ko'nikmalar olinadi:

So'rovlarni rejalashtirish va bajarilishini mexanizmini to'liq bilish.
Biror serverni samaradorlikka doir parametrlarini sozlash.
Muammoli so'rovlarni qidirish va ularni maqbullashtirish.

Mavzular:

1. "Havo transporti" demo bazasi
2. So'rovlarni bajarilishi
3. Ketma-ket kirish
4. Indeksli kirish
5. Bit xaritasi bo'yicha skanerlash
6. Ichma-ich sikl yordamida bog'lash
7. Xeshlash yordamida bog'lash
8. Birlashtirish yordamida bog'lash
9. Statistika
10. Profil yaratish
11. Maqbullashtirish usullari

O'quv kursi materiallari mustaqil o'rganish uchun postgrespro.ru/education/courses/QPT manzilida mavjud.

PGPRO. Postgres Pro Enterprise imkoniyatlari

Davomiylik: 3 kun

Talab etilgan bilimlar:

Unix OT bilan tanish bo'lish.
SQL bo'yicha mustahkam bilimlar.
PL/pgSQL tilini bilish foydali ammo zarur emas.
PostgreSQL bilan DBA1, DBA2, QPT (administratorlar uchun) yoki DEV1, DEV2, QPT (dasturchilar uchun) kurslar hajmida tanish bo'lish.

Qanday ko'nikmalar olinadi:

Postgres Pro Enterprise qo'shimcha imkoniyatlarini ishlatish.

Mavzular:

1. Tahrirlar va imkoniyatlar
2. O'rnatish, sozlash, yangilanish
3. Tranzaksiyalarni boshqarish
4. CFS — siqilgan fayllar tizimi
5. So'rovlarni maqbullashtirish
6. Moslashtirilgan maqbullashtirish
7. Samaradorlik tahlili
8. pgpro_pwr yuklamasi boy'icha hisobotlar
9. Foydalanuvchilar profillari
10. Audit

11. Topshiriqlar rejalashtiruvchisi
12. Zaxira uchun nusxalash — 1
13. Zaxira uchun nusxalash — 2
14. Zaxira uchun nusxalash — 3
15. Sinxron klaster multimaster

O'quv kursi materiallari mustaqil o'rganish uchun postgrespro.ru/education/courses/PGPRO manzilida mavjud.

Professional sertifikatlash

2019 yilda ishga tushirilgan professional sertifikatlash dasturi nafaqat mutaxassislar uchun, balki ish beruvchilar uchun ham foydalidir. Sertifikat egalariga ish qidirishda va ish haqi darajasini muhokama qilishda qo'shimcha afzalliklar berilishi mumkin. Bundan tashqari, bu mustaqil baholash tizimidan foydalanib, o'z bilim darajasini tasdiqlash imkoniyatidir.

Ish beruvchilar uchun dastur yangi mutaxassislarni topishni osonlashtiradi va mavjud xodimlarning PostgreSQL bo'yicha bilim darajasini tekshirish imkonini beradi. Bu dastur – xodimlarni o'qitishga yuborishda olingan bilimlarning sifatini nazorat qilish, shuningdek, hamkor kompaniyalar va xizmat ko'rsatuvchi provayderlar xodimlarining malakasiga ishonch hosil qilish imkoniyatini taqdim etadi.

Hozirda sertifikatlash faqat ma'lumotlar bazasi administratorlari uchun mavjud, ammo kelgusida dastur ilova dasturchilariga ham kengaytirilishi rejalashtirilgan.

Sertifikatlash uch darajani nazarda tutadi, har bir darajaga erishish uchun bir qator testlardan o'tish kerak bo'ladi.

"Professional" darajasi quyidagi sohalardagi bilimlarni tasdiqlaydi:

- PostgreSQL arxitekturasi haqida umumiy tushunchalar;
- Serverni o'rnatish variantlari, psql bilan ishlash ko'nikmalari, konfiguratsiya sozlamalarini boshqarish;
- Ma'lumotlarni mantiqiy va fizik darajada tashkil etish;
- Foydalanuvchilarni va kirishni boshqarish;
- Ma'lumotlar bazasini zaxira nusxalash va replikatsiya qilish haqida umumiy tushunchalar.

Sertifikat olish uchun DBA1 kursi bo'yicha testni mu-vaffaqiyatli topshirish kerak. Barcha yangi nomzodlarga PostgreSQL 13 versiyasi bo'yicha test topshirishni tavsiya qilamiz.

"Ekspert" darajasi quyidagi sohalardagi qo'shimcha bilimlarni tasdiqlaydi:

- PostgreSQLning ichki tuzilishi;
- Serverni monitoring qilish va sozlash, texnik xizmat ko'rsatish vazifalarini bajarish;
- Ishlash samaradorligini optimallashtirish, so'rovlarni sozlash masalalarini hal qilish;
- Zaxira uchun nusxalashni bajarish;
- Turli ishlash senariylari uchun fizik va mantiqiy replikatsiyani sozlash.

Sertifikatni olishning ikki yo'li mavjud:

- PostgreSQL 13 bo'yicha "Professional" darajasidagi sertifikatga ega bo'lish va DBA2-13, DBA3-13, QPT-13 testlarini (istalgan tartibda) muvaffaqiyatli topshirish;

- PostgreSQL 10 bo‘yicha “Ekspert” darajasidagi sertifikatga ega bo‘lish va bitta o‘tish testi, ya‘ni Ekspert 10–13 testini muvaffaqiyatli topshirish.

“Master” darajasi qo‘shimcha ravishda PostgreSQLni boshqarish bo‘yicha amaliy ko‘nikmalarni tasdiqlaydi.

Sertifikat olish uchun “Ekspert” darajasidagi sertifikatga ega bo‘lish va amaliy testni muvaffaqiyatli topshirish kerak. Ushbu sertifikat turi hozirda ishlab chiqilmoqda.

postgrespro.ru/user saytida ro‘yxatdan o‘ting va shaxsiy kabinetda testdan o‘tish uchun yoziling.

Testlarni muvaffaqiyatli topshirish uchun:

- Tegishli kurslarning materiallarini yaxshi bilish va kurslarda keltirilgan havolalar orqali hujjatlar bo‘limlari bilan tanish bo‘lish;
- PostgreSQL bilan psql muhitida amaliy ishlash ko‘nikmalariga ega bo‘lish kerak.

Testdan o‘tish jarayonida kurs materiallari va PostgreSQL hujjatlari mavjud, ammo boshqa manbalardan foydalanish taqiqlanadi.

Navbatdagi darajaga erishish sertifikat bilan tasdiqlanadi. Sertifikat muddatsizdir, lekin ma‘lum bir server talqiniga bog‘langan va ushbu talqin bilan birga eskiradi, shuning uchun bir necha yil o‘tgach, PostgreSQLning dolzarb talqini bo‘yicha testdan o‘tish zarurati paydo bo‘lishi mumkin.

Sertifikatlash dasturi haqida batafsil ma‘lumotni postgrespro.ru/education/cert saytida o‘qing.

Ta‘lim beruvchi tashkilotlar bilan hamkorlik

Kompaniyamizning eng muxim faoliyat yo‘nalishlaridan biri ma‘lumotlar bazasi boshqaruv tizimlari sohasida kadrlar tayyorlashdir. Kelajakdagi mutaxassislarni tayyorlashni ta‘lim davridan boshlash kerak, bu esa faqat oliy o‘quv yurtlari, kollejlar va maktablar bilan hamkorlikda amalga oshirilishi mumkin.

Akademik litsenziya

Biz tegishli tashkilotlar uchun Postgres Pro Standard MBBT-ning akademik litsenziyasini bepul taqdim etamiz. Ariza yuborish uchun academy@postgrespro.ru manziliga murojaat qiling.

Talabalar uchun amaliyotlar

Biz oliy o‘quv yurtlarida amaliyot o‘tash uchun o‘quv laboratoriyalarini tashkil qilamiz. Kompaniyamiz amaliyotchilarga PostgreSQL bilan bevosita bog‘liq diplom mavzularini taklif etadi va himoya qilishda yordam beradi.

Musobaqalar

Kompaniya xakatonlar tashkil etishda ishtirok etadi va maktab o‘quvchilari hamda talabalar uchun turli musobaqalar o‘tkazadi. Ular orasida:

- “Postgres Pro DBMS” musobaqasi “IT-Planeta” olimpiadasida: it-planet.braim.org/2024/postgresql.html;
- Butunrossiya ochiq kodli loyihalar musobaqasi: foss.kruzhok.org.

Kompaniya ishtirokida o'tkazilgan tanlovlar g'oliblari va final ishtirokchilari, shuningdek, hamkor universitetlarning o'qituvchilari professional sertifikatlash dasturi bo'yicha testlarni bepul topshirishlari mumkin. Promokod olish uchun certification@postgrespro.ru manziliga yozing va tasdiqlovchi hujjatni qo'shing.

Oliy o'quv yurtlari uchun kurslar

Biz kompaniyamiz va yetakchi universitetlarning tajribali o'qituvchilari hamkorligining natijasi bo'lgan bir nechta o'quv kurslarini taklif etamiz. Barcha kurslar ta'lim faoliyatida foydalanish uchun ochiq. O'qituvchilarga taqdim etiladigan materiallar: o'quv qo'llanmalari, taqdimot slaydlari va ma'ruza videoyozuvlari, shuningdek, boshqa o'quv materiallari postgrespro.ru/education/university manzilida mavjud.

Bizning ishtirokimizda ishlab chiqilgan kurslar quyidagi universitetlarda o'qitiladi: M. V. Lomonosov nomidagi Moskva davlat universiteti, Oliy iqtisodiyot maktabi, Moskva aviatsiya instituti, M. F. Reshetnev nomidagi Sibir davlat fan va texnologiya universiteti, Sibir federal universiteti. Agar siz universitet vakili bo'lsangiz va o'quv rejangizga ma'lumotlar bazalari bo'yicha kurslarni kiritishga qiziqsangiz, biz bilan bog'laning.

Shuningdek, biz PostgreSQLdan foydalangan holda yangi mualliflik kurslarini ishlab chiqishga tayyor bo'lgan o'qituvchilarni hamkorlikka taklif qilamiz. Biz o'z navbatida qo'llab-quvvatlaymiz, maslahat beramiz, qo'lyozmalarni tahrirlaymiz va nashrga tayyorlaymiz, mualliflar uchun mamlakatimiz yetakchi universitetlarida ochiq ma'ruzalar tashkil qilamiz.

Kurs tinglovchilari oldindan tayyorgarliksiz PostgreSQL tizimi nima ekanligini tushunib olishlari va u bilan ishlashni o'rganishlari mumkin. SQL tilida oddiy so'rovlarni ishlab chiqishdan boshlanib, tinglovchilar asta-sekin murakkab tuzilmalarga o'tishadi, tranzaktsiya tushunchasi va ishlashni optimallashtirish bilan tanishadilar.

Kursning asosi "PostgreSQL. SQL Tilining Asoslari" o'quv qo'llanmasiga asoslanadi.

Mundarija:

Моргунов Е. П.
PostgreSQL. Основы языка SQL: учеб. пособие – СПб.: БХВ-Петербург, 2018. – 336 с.

ISBN 978-5-9775-4022-3 (qog'oz nashri)
ISBN 978-5-6041193-2-7 (elektron nashri)

Elektron ko'rinishdagi kitob bizning saytimizda mavjud: postgrespro.ru/education/books/sqlprimer.

Kurs 36 soatlik ma'ruza va amaliy mashg'ulotlardan iborat. Ushbu kurs muallif tomonidan bir necha yildan beri Moskva va Krasnoyarskdagi yetakchi universitetlarda doimiy ravishda o'qitilib kelinmoqda. Kurs materiallari postgrespro.ru/education/university/sqlprimer manzilida mavjud.

Yevgeniy Pavlovich Morgunov, texnika fanlari nomzodi, akademik M. F. Reshetnev nomidagi Sibir davlat fan va texnologiya universiteti informatika va hisoblash texnikasi kafedrasi dotsenti.

Yevgeniy Pavlovich Morgunov Krasnoyarskda yashaydi. 2000-yilda universitetga o'tishdan oldin o'n yildan ortiq vaqt davomida dasturchi bo'lib ishlagan, shu jumladan bank uchun dasturiy tizim ishlab chiqish bilan shug'ullangan. 1998-yilda PostgreSQL MBBT bilan tanishgan. Ta'lim jarayonida ochiq va bepul dasturiy ta'minotdan foydalanish tarafdori. Uning tashabbusi bilan "Dasturlash texnologiyasi" fanini o'rganish jarayonida FreeBSD operatsion tizimi va PostgreSQL ma'lumotlar bazasi boshqaruv tizimi qo'llanila boshlandi. Xalqaro muhandislik pedagogikasi jamiyati (IGIP) a'zosi. PostgreSQLdan ta'limda foydalanish tajribasi yigirma yildan ortiq.

Zamonaviy universitet kursi bo'lib, u chuqur nazariy tashkil etuvchilarni ma'lumotlar bazasining boshqaruv tizimlarini qo'llash va loyihalashning dolzarb amaliy jihatlari bilan birlashtiradi.

Birinchi qism ma'lumotlar bazasi boshqaruv tizimlari haqida asosiy ma'lumotlarni o'z ichiga oladi: relyatsion ma'lumotlar modeli, SQL tili, tranzaksiyalarni qayta ishlash.

Ikkinchi qismda MBBTning ishlashiga asos bo'lgan texnologiyalar va ularning rivojlanish tendensiyalari batafsil ko'rib chiqiladi. Ba'zi mavzular chuqurroq darajada takroran o'rganiladi.

Новиков Б. А.
Основы технологий баз данных: учеб. пособие / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева; под ред. Е. В. Рогова. – 2-е изд. – М.: ДМК Пресс, 2020. – 582 с.

ISBN 978-5-97060-841-8 (qog'oz nashri)
ISBN 978-5-6041193-5-8 (elektron nashri)

Mundarija:

Kitobning elektron ko'rinishi saytda mavjud: postgrespro.ru/education/books/dbtech.

Kurs 24 soatlik ma'ruza va 8 soatlik amaliy mashg'ulotlardan iborat. Kurs Boris Asenovich Novikov tomonidan M. V. Lomonosov nomidagi Moskva davlat universiteti VMK fakultetida o'qitilgan. Kurs materiallari postgrespro.ru/education/university/dbtech manzilida mavjud.

Boris Asenovich Novikov, fizika-matematika fanlari doktori, Sankt-Peterburgdagi Oliy iqtisod maktabi informatika departamenti professori.

Ilmiy qiziqishlari asosan ma'lumotlar bazasi boshqaruv tizimlarini va ularning ilovalarini loyihalash, ishlab chiqish va qo'llashning turli jihatlari, shuningdek, katta ma'lumot oqimlarini qayta ishlash va tahlil qilish uchun taqsimlangan masshtablanuvchi tizimlar bilan bog'liq.

Gorshkova Yekaterina Aleksandrovna, fizika-matematika fanlari nomzodi.

Yuqori yuklangan va ma'lumotlardan intensiv foydalanadigan ilovalarni loyihalash bo'yicha mutaxassis. Ilmiy qiziqishlariga mashinani o'rganish, oqim ma'lumotlarini tahlil qilish va axborot qidiruvi kiradi.

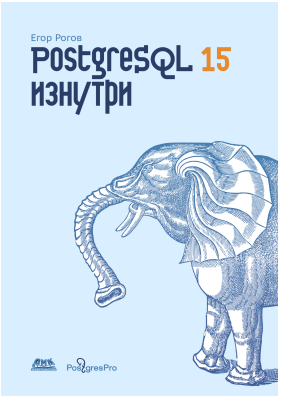
Grafeeva Natalya Genrixovna, fizika-matematika fanlari nomzodi, Sankt-Peterburg davlat universiteti axborot-tahliliy tizimlar kafedrasi dotsenti.

Ilmiy qiziqishlari ma'lumotlar bazalari, axborot qidiruvi, katta ma'lumotlar va intellektual ma'lumotlar tahlili bilan bog'liq. Axborot tizimlarini ishlab chiqish, loyihalash va qo'llab-quvvatlash, shuningdek, o'quv kurslarini ishlab chiqish va o'qitish bo'yicha katta tajribaga ega.

Kitoblar

PostgreSQL ichkaridan qaralganda

Ushbu kitob, ma'lumotlar bazasi bilan qora quti sifatida ishlashdan qoniqmaydiganlar uchun. Kitob PostgreSQLdan foydalanishda biroz tajribaga ega bo'lgan o'quvchilar uchun mo'ljallangan. Bu kitob boshqa ma'lumotlar bazasi boshqaruv tizimlari bilan yaxshi tanish bo'lgan, lekin PostgreSQLga o'tayotgan va farqlarni tushunishni xohlaydiganlar uchun ham foydali bo'ladi.



Kitobda siz tayyor retseptlarni topa olmaysiz, lekin ichki mexanizmlarni tushunish sizga boshqalarning tajribasini tanqidiy qayta ko'rib chiqish va mustaqil xulosalar chiqarish imkonini beradi. Muallif PostgreSQL tuzilishining tafsilotlarini tushuntiradi va qanday qilib tajribalar o'tkazishni va eskirgan ma'lumotlarni mustaqil ravishda tekshirishni ko'rsatadi.

Рогов Е. В.
PostgreSQL изнутри. – М.: ДМК Пресс, 2022. – 660 с.

ISBN 978-5-93700-122-1 (qog'oz nashri)
ISBN 978-5-6041193-9-6 (elektron nashri)

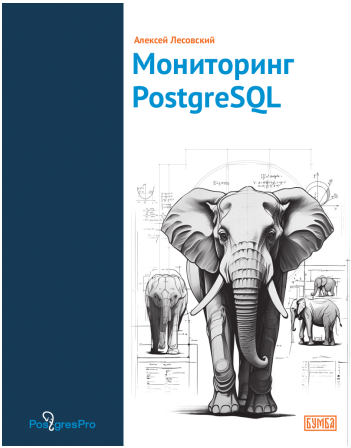
Egor Rogov 2015 yildan beri Postgres Professional kompaniyasining ta'lim bo'limida ishlaydi: o'quv kurslarini ishlab chiqadi va o'qiydi, maqolalar nashr qiladi, kitoblar yozadi va tahrir qiladi.

Mundarija:

Elektron ko'rinishidagi kitob bizning saytimizda mavjud: postgrespro.ru/education/books/internals.

PostgreSQL monitoring

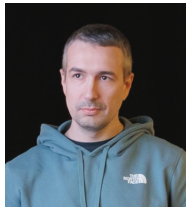
PostgreSQL monitoringi administratorning ishining muxim qismidir. Kitob ushbu keng mavzuni har tomonlama qamrab oladi, mavjud vositalar haqida ma'lumotnoma materiallarini, ularni qo'llashning amaliy usullarini va olingan ma'lumotlarni talqin qilish usullarini birlashtiradi.



PostgreSQLning ichki tuzilishini va uning monitoring xususiyatlarini bilish, ushbu kitobdan olingan bilimlar, uzoq muddatli istiqbolda MBBTni samarali ishlatishga va paydo bo'ladigan boshqaruv vazifalarini muvaffaqiyatli hal qilishga yordam beradi.

Лесовский А. В.
Мониторинг PostgreSQL. – М.: Бумба, 2024. – 247 с.

ISBN 978-5-907754-42-3



Aleksey Lesovskiy — ma'lumotlar bazasining professional administratori, tizim administratori, dasturchi, devops-muhandis. U qariyb 20 yildan beri katta va murakkab tizimlarni ekspluatatsiya qilish, dasturiy ta'minotni loyihalash va ishlab chiqish bilan shug'ullanadi.

Mundarija:

Kitob bilan birga Docker muhiti taklif etiladi, unda barcha misollarni qayta tiklash va monitoring bilan bog'liq har qanday eksperimentlarni o'tkazish mumkin. Kitob ma'lumotlar bazasi administratorlari, tizim administratorlari, ishonchlilik mutaxassislari va ishlash samaradorligini optimallashtirish bilan qiziqqan barcha uchun foydali bo'ladi.

Elektron shaklda kitob bizning saytimizda mavjud: postgrespro.ru/education/books/monitoring.

Maʻlumotlar bazalariga yoʻriqnoma

Kitob zamonaviy maʻlumotlar bazasi boshqaruv tizimlarining arxitektura prinsiplari, shuningdek, ularda qoʻllaniladigan algoritmlar va maʻlumotlar tuzilmalarini tushuntiradi. Oʻxshash funksionallikdagi platformalarning bir xil yondashuvlarining amalga oshirilishini taqqoslashga alohida eʻtibor qaratilgan.

Ushbu kitobni uch oy davomida “ITga kirish” kurslarida olingan bilimlardan qoniqmagan barcha oʻqishi kerak. Kitob amaliy bilimlarni taqdim etadi va umumiy qoidalarni tushunishda mustahkam asoslar beradi. Kitob axborot tizimlari arxitektorlariga va yetakchi dasturchilarga moʻljallangan. Boshqacha qilib aytganda, u elitaga va ushbu darajaga chiqmoqchi boʻlganlarga qaratilgan.

Комаров В. И.
Путеводитель по базам данных. – М.: ДМК Пресс, 2024. – 520 с.

ISBN 978-5-93700-287-7

Elektron shaklda kitob bizning saytimizda mavjud: postgrespro.ru/education/books/dbguide.

Vladimir Komarov – keng profilli IT mutaxassis: dasturchi, ma'lumotlar bazasi administrator, ma'lumot va infratuzilma arxitektori, o'qituvchi va ozgina evangelist.

Mundarija:

XI Galaktika bo‘yicha yo‘riqnoma

Yangiliklar va muhokamalar

Har qanday qiziquvchi yangiliklar bilan tanishib, PostgreSQL-ning yangi imkoniyatlari va kelgusi nashri haqida ma‘lumot olib, umuman olganda, voqealardan xabardor bo‘lishi mumkin.

Ko‘plab qiziqarli va foydali materiallar turli tematik bloglarda chop etiladi. Ingliz tilidagi maqolalar bilan to‘liq tanishish uchun planet.postgresql.org sayti-aggregatoridan foydalanish qulay, rus tilidagi maqolalar esa (shu jumladan, bizning kompaniyamiz tomonidan ham) Xabrda chop etiladi: habr.com/hub/postgresql. Youtube kanallariga ham e‘tibor bering: youtube.com/RuPostgres va youtube.com/PostgresTV.

Shuningdek, wiki.postgresql.org viki-loyihasi mavjud bo‘lib, u yerda odatiy savollarga javoblar, o‘quv materiallari va maqolalar, sozlash va optimallashtirish bo‘yicha ma‘lumotlar, turli MBBT lardan migratsiya xususiyatlari va boshqalar mavjud. Ushbu materiallarning bir qismi rus tilida ham mavjud: wiki.postgresql.org/wiki/Russian. Har kim ingliz tilidagi maqolani tarjima qilib, hamjamiyatga yordam berishi mumkinligini unutmang.

Deyarli 12 000 rus tilida so‘zlashuvchi foydalanuvchilar “pgsql – PostgreSQL” telegram kanaliga obuna bo‘lgan (t.me/pgsql); ayni paytda bu yerda eng faol va yordam berishga tayyor hamjamiyat yig‘ilgan.

Deyarli 600 o‘zbek tilida so‘zlashuvchi foydalanuvchilar “pgsql – PostgreSQL Uzbekistan” telegram kanaliga obuna bo‘lgan (t.me/pgsql_uzbekistan); ayni paytda bu yerda eng faol va yordam berishga tayyor hamjamiyat yig‘ilgan.

Profilli saytlarda qo‘shimcha savollar berish mumkin, masalan, stackoverflow.com (ingliz tilida), ru.stackoverflow.com saytlarda (rus tilida, har qanday holatda ham postgresql tegini qo‘shishni unutmang).

Postgres Professional o‘zining shaxsiy maqolalarini ushbu adresda jamlab boradi postgrespro.ru/blog.

Pochta ro‘yxatlari

Blogda kimdir maqola yozishini kutish shart emas, pochta ro‘yxatiga obuna bo‘lish mumkin. PostgreSQL ishlab chiquvchilari barcha masalalarni faqat elektron pochta orqali muhokama qilish an‘anasiga ega.

Barcha pochta ro‘yxatlarining to‘liq ro‘yxati postgresql.org/list manzilida mavjud. Ular orasida:

- pgsql-hackers (odatda “hackers” deb ataladi) – rivojlanish bilan bog‘liq barcha masalalar bo‘yicha asosiy ro‘yxat;
- pgsql-general umumiy savollarni muhokama qilish uchun;
- pgsql-bugs topilgan xatolar haqida xabar berish uchun;

- pgsql-docs hujjatlarni muhokama qilish uchun;
- pgsql-translators tarjimonlar uchun;
- pgsql-announce yangi mahsulot versiyalari chiqishi haqidagi yangiliklar uchun...

...va ko'pgina boshqalar.

Ushbu ro'yxatlardan biriga obuna bo'lganingizda, siz muntazam ravishda elektron pochta orqali xabarlarni olasiz va istasangiz, muhokamalarda ishtirok etishingiz mumkin. Yana bir variant — postgresql.org/list yoki bizning kompaniya saytimizdagi postgrespro.ru/listda xabarlar arxivini o'qish.

Commitfest

Yana bir ko'p vaqt sarflamasdan voqealardan xabardor bo'lish usuli — commitfest.postgresql.org saytiga nazar tashlash. Ushbu tizimda davriy ravishda "oynalar" ochiladi, bu davrda ishlab chiquvchilar o'z patchlarini ro'yxatdan o'tkazishlari kerak. Masalan, 2023-yil 1-martdan 2023-yil 31-martgacha bo'lgan oyna PostgreSQL 16 versiyasiga tegishli edi, undan keyingi oyna esa 2023-yil 1-iyuldan 2023-yil 31-iyulgacha bo'lib, keyingi talqinga tegishli edi. Bu yangi talqin chiqishidan taxminan olti oy oldin yangi imkoniyatlarni qabul qilishni to'xtatish va kodni barqarorlashtirish uchun amalga oshiriladi.

Patchlar bir necha bosqichdan o'tadi: ular ko'rib chiqiladi va taqriz natijalariga ko'ra tuzatiladi, keyin esa qabul qilinadi yoki keyingi oynaga o'tkaziladi yoki — muvaffaqiyatsizlik bo'lsa — rad etiladi.

Shu tariqa, siz yangi talqinga allaqachon kiritilgan va kiritilishi rejalashtirilayotgan imkoniyatlar haqida bilib olishingiz mumkin.

Konferensiyalar

Moskva, Sankt-Peterburg va Rossiyaning boshqa shaharlarida muntazam ravishda yuzlab PostgreSQL foydalanuvchilari va ishlab chiquvchilarni yig'adigan yirik xalqaro **PGConf** (pgconf.ru) konferensiyasi o'tkaziladi.

Bundan tashqari, mamlakatning turli shaharlarida ma'lumotlar bazalari, shu jumladan PostgreSQL yo'nalishini qamrab oladigan kengroq mavzular bo'yicha konferensiyalar o'tkaziladi. Shulardan ba'zilarini qayd etamiz:

CodeFest – Novosibirskda (codefest.ru);

HighLoad++ – Moskva va boshqa shaharlarda (highload.ru).

Albatta, konferensiyalar boshqa mamlakatlarda ham o'tkaziladi. Ularning eng yiriklari:

PGCon – Ottavada (pgcon.org);

Yevropacha **PGConf Europe** (pgconf.eu).

Konferensiyalardan tashqari, norasmiy uchrashuvlar, shu jumladan, onlayn uchrashuvlar ham o'tkaziladi.

XII Kompaniya haqida

Postgres Professional kompaniyasi 2015 yilda tashkil topgan bo'lib, uning tarkibiga jahon hamjamiyati tomonidan PostgreSQL rivojlanishiga qo'shgan hissasi e'tirof etilgan asosiy rossiyalik dasturchilar kirgan. Unda MBBT sohasi bo'yicha malakali mahalliy kadrlar tayyorlanadi. Hozirgi kunda kompaniyada taxminan 250 dasturchi, arxitektor, muhandis va boshqa mutaxassislar ishlaydi.

Postgres Professional PostgreSQL asosida qurilgan Postgres Pro tizimining bir nechta talqinlarini chiqaradi, MBBT yadro darajasida va kengaytmalarda ishlab chiqishlar amalga oshiradi, dasturiy tizimlarni loyihalash va qo'llab-quvvatlash hamda PostgreSQLga migratsiya qilish bo'yicha xizmatlar ko'rsatadi.

Kompaniya ta'lim faoliyatiga katta e'tibor qaratadi, Moskvada har yili o'tkaziladigan eng yirik xalqaro konferentsiya PgConf.Russiani tashkil qiladi va butun dunyo bo'ylab konferentsiyalarda ishtirok etadi.

Aloqa ma'lumotlari:

117036, г. Москва, ул. Дмитрия Ульянова, д. 7А

+7 495 150-06-91

info@postgrespro.ru

Postgres Pro MBBT

Postgres Professional kompaniyasining Postgres Pro – bu erkin tarqatiladigan PostgreSQL ma'lumotlar bazasi boshqaruvi tizimiga asoslangan va korporativ mijozlarning talablari bo'yicha ishlab chiqilgan Rossiyaning tijorat MBBT hisoblanadi. U Rossiya dasturiy ta'minot reestriga kiritilgan.

Postgres Pro Standard PostgreSQLning barcha funksional imkoniyatlarini o'z ichiga oladi va turli kengaytmalar va patchlar bilan to'ldirilgan, jumladan hali hamjamiyat tomonidan qabul qilinmaganlarini ham. Mijoz navbatdagi PostgreSQL chiqarilishini kutmasdan foydali funksional imkoniyatlardan foydalanish va ishlash tezligini oshirish imkoniyatiga ega bo'ladi.

Postgres Pro Enterprise – bu chuqur qayta ishlangan MBBT talqini bo'lib, yuqori ishonchliligi va oshirilgan unumdorligi tufayli jiddiy sanoat vazifalarini hal qilish uchun mos keladi.

Axborotni himoya qilish uchun zarur vositalar bilan to'ldirilgan har ikkala Postgres Pro talqini **FSTEK sertifikatidan** o'tgan.

Postgres Proning har qanday talqinidan foydalanish uchun litsenziya sotib olish kerak. Sizni qiziqtirgan MBBT talqinini sinovdan o'tkazish, imkoniyatlarini o'rganish va dasturiy ta'minotni ishlab chiqish uchun bepul olishingiz mumkin.

Ta'lim muassasalariga Postgres Pro Standard MBBT uchun bepul akademik litsenziya taqdim etiladi.

Postgres Pro versiyalari imkoniyatlari va farqlari haqida batafsil ma'lumotni quyidagi veb-sahifada o'qishingiz mumkin: postgrespro.ru/products/postgrespro.

Postgres Pro Enterprise Manager

Postgres Pro Enterprise Manager (PPEM) – bu Postgres Pro Enterprise MBBT uchun integratsiyalashgan boshqaruv paneli.

PPEM yagona konsolni taqdim etadi, oddiy monitoring va boshqaruv interfeysi bilan birga, asosiy adminstratorlik harakatlarini brauzer oynasidan bajarish imkonini beradi, barcha serverlarga va ma'lumotlar bazalariga markazlashgan kirishni ta'minlaydi. PPEM ma'lumotlar bazasi xizmat ko'rsatish guruhining mahsuldorligini oshiradi va kundalik ma'muriy vazifalarni bajarishni osonlashtiradi.

Batafsil: postgrespro.ru/products/PPEM.

Postgres Pro Backup Enterprise

Postgres Pro Backup Enterprise (`pg_probackup`) – bu PostgreSQL ma'lumotlar bazasi klasterlarini zaxira qilish va tiklashni boshqarish uchun mo'ljallangan utilita.

Ushbu utilita markazlashtirilgan nusxa katalogi bilan ishlaydi, u mahalliy yoki masofaviy, jumladan, S3 xotirasida joylashishi mumkin.

Ushbu utilita sahifa darajasidagi inkremental zaxira nusxalari, zaxira nusxalarini saqlash siyosatlari, ma'lumotlar yaxlitligini tekshirish va belgilangan vaqtga tiklash imkoniyatlarini qo'llab-quvvatlaydi. Shuningdek, u parallel zaxiralash va tiklash, ma'lumotlarni siqish va CFS fayl tizimi bilan ishlashni ta'minlaydi.

Batafsil: postgrespro.ru/products/pg_probackup.

Shardman

Shardman – bu PostgreSQL asosida yaratilgan taqsimlangan relyatsion ma'lumotlar bazasi boshqaruv tizimi bo'lib, Postgres Pro MBBTning kengaytirilgan imkoniyatlariga ega.

Shardman PostgreSQL bilan mos keladi, ma'lumotlarning izolyatsiyasi va izchilligiga qat'iy kafolatlar beradi, gorizontal masshtablashni ta'minlaydi va o'rnatilgan nosozliklarga chidamlilikka ega.

Batafsil: postgrespro.ru/products/shardman.

Karyera

Biz Postgres ekotizimi mahsulotlarini ishlab chiqishga va ochiq dasturiy ta'minotni rivojlantirishga tayyor bo'lgan mutaxassislarni faol izlayapmiz.

Postgres Professional jamoasida Rossiyadagi ko'pgina Postgres mutaxassislari ishlaydi. Biz PostgreSQL yadrosi va uning kengaytmalari darajasida ishlab chiqish bilan shug'ullanamiz, monitoring va MBBT boshqaruvi uchun ekotizim mahsulotlarini yaratamiz. Bizning mutaxassislarimiz DBaaS xizmatini ishlab chiqadi, texnik audit va ma'lumotlarni migratsiya qiladi, texnik yordam ko'rsatadi, tadqiqotlar olib boradi va innovatsion mahsulotlar yaratadi.

Biz tizim dasturchilari, backend va frontend dasturchilari, DevOps mutaxassislari, QA muhandislari, ma'lumotlar bazasi administratorlari va boshqa yo'nalishlardagi mutaxassislar bilan hamkorlik qilishga ochiqmiz. Batafsil ma'lumotni bizning karyera saytimizda olishingiz mumkin: career.postgrespro.ru.

Xizmatlar

Postgres MBBT uchun nosozliklarga chidamli yechimlar

Yuqori yuklamali, yuqori samarali va nosozliklarga chidamli sanoat tizimlarini loyihalashtirish va yaratishda ishtirok etish, konsalting xizmatlari. Postgres MBBTni joriy etish va konfiguratsiyani optimallashtirish.

Vendorlik texnik qo'llab-quvvatlash

Postgres Pro va PostgreSQL uchun 24x7 texnik qo'llab-quvvatlash. Monitoring, ishlash qobiliyatini tiklash, kutilmagan holatlarni tahlil qilish, samaradorlikni oshirish, MBBT va kengaytmalardagi xatolarni tuzatish.

Ilovalar tizimlarini Postgres MBBTga ko'chirish

Boshqa MBBTlardan Postgresga migratsiya qilishning murakkabligini baholash. Yangi yechim arxitekturasini ishlab chiqish va zaruriy yaxshilanishlarni amalga oshirish. Ilovalar tizimlarini Postgres MBBTga ko'chirish va migratsiya jarayonida qo'llab-quvvatlash.

Postgres bo'yicha o'qitish.

Ma'lumotlar bazasi administratorlari, dasturchilar va ilova tizimlari arxitektorlarini Postgres MBBTning xususiyatlari va uning afzalliklaridan samarali foydalanishga o'rgatish.

MBBT auditi

MBBT holatining ekspert bahosi. Postgres asosidagi tizimlarning axborot xavfsizligi auditi.

Xizmatlarning to'liq tavsifi: postgrespro.ru/services.

Luzanov Pavel Veniaminovich
Rogov Egor Valerevich
Lyovshin Igor Viktorovich

Postgres. Birinchi tanishuv

Muharrir: Pyetr Lagutkin
Muqova dizayneri: Eteri Bejashvili
Tarjimon: Faxriddin Baltayev, Qosim Xudaybergenov

10-nashr, qayta ishlangan ba to'ldirilgan

postgrespro.ru/education/books/introbook

Москва, Постгрес Профессиональный, 2024

ISBN 978-5-6045970-6-4