**ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ**

БД – База даних

СКБД – Система керування базами даних

SQL – Structured Query Language

CRM – Система управління відносинами з клієнтами

VCL – Бібліотека візуальних компонентів

IT – Інформаційні технології

ВДТ – Відеодисплейні термінали

ПК – Портативний комп’ютер

ЕОМ – Електронна обчислювальна машина

**ВСТУП**

Сьогодні неможливо представити підприємство, що не мають комп’ютерів. В наш час автоматизувати облік стало не тільки модним, але й необхідним для підвищення діяльності підприємства в цілому.

При впровадженні комп'ютерних інформаційних облікових систем на підприємствах, що базуються на сучасних інформаційних технологіях, підприємство стикається з великою кількістю переваг. Проблема створення або впровадження цих систем на підприємстві залишається однією з найскладніших, не лише в Україні, а й у світі.

Сучасні підприємства вимагають найновіших методів управління. Ухвалення управлінських рішень в умовах постійно змінного, динамічного середовища вимагає не тільки аналізу, оцінки і прогнозування внутрішнього розвитку підприємства, але і забезпечення результатами фінансово-господарської діяльності підприємства, що, у свою чергу, висуває високі вимоги до інформаційного забезпечення управління підприємством.

В даний час на ринку комп'ютерних програм є універсальні аналітичні програми і спеціальні. Більшість підприємств віддають перевагу універсальним комп'ютерним програмам внаслідок їх адаптації до особливостей бухгалтерського обліку в різних галузях. Але не всім підприємствам підходять ці програмі. Тому дуже часто підприємства наймають програміста для написання програми, яка буде відповідати вимогам цього підприємства.

Метою даної роботи є написання програми ведення обліку та грошових транзакцій для компанії ТімТек. Ця програма дозволить вести облік клієнтів компанії, проектів які там розробляються, автоматизувати розрахунок витрат та прибутку компанії, а також нарахування зарплати працівникам. Головними відмінностями програми від інших в тому, що вона буде проста у використанні, в ній поєднані як функції ведення простого обліку так і бухгалтерського, також в ній не буде нічого лишнього, так як вона написана спеціально для потреб компанії ТімТек.

**1 АНАЛІЗ ТА ДОСЛІДЖЕННЯ ВЕДЕННЯ ОБЛІКУ КЛІЄНТІВ ТА ГРОШОВИХ ТРАНЗАКЦІЙ ПІДПРИЄМСТВА**

* 1. **Теоретичні відомості про види обліку**

Облік — віддзеркалення господарської або іншої діяльності підприємства на підставі документів в різних вимірниках (кількісних і (або) якісних). Облік є складовою частиною управління економічними процесами і об'єктами, суть обліку полягає у фіксації їх стану і параметрів, зборі і накопиченні відомостей про економічні об'єкти і процеси, віддзеркалення цих відомостей в облікових відомостях. Розрізняють аналітичний, синтетичний, бухгалтерський, бюджетний облік. Облік може здійснюватися в поточних і незмінних (зіставних) цінах, а також в іноземній валюті.

Бухгалтерський облік — процес виявлення, вимірювання, реєстрації, накопичення, узагальнення, зберігання та передачі інформації про діяльність підприємства зовнішнім та внутрішнім користувачам для прийняття рішень.

Об'єктами бухгалтерського обліку є майно організацій, їх зобов'язання і господарські операції, здійснювані організаціями в процесі їх діяльності.

Основним завданням бухгалтерського обліку є формування повної і достовірної інформації про діяльність організації і її майнове положення, необхідною внутрішнім користувачам бухгалтерської звітності — керівникам, засновникам, учасникам і власникам майна організації, а також зовнішнім — інвесторам, кредиторам і іншим користувачам бухгалтерської звітності.

Ту частину системи бухгалтерського обліку, що забезпечує потребу керівництва в інформації, називають управлінським обліком. Управлінський облік – це процес виявлення, вимірювання, накопичення, аналізу, підготовки, інтерпретації та передачі інформації, що використовується управлінською ланкою для планування, оцінки i контролю всередині організації та для забезпечення відповідного підзвітного використання ресурсів.

Стандартизація бухгалтерського обліку — процес розробки та поступового застосування єдиних вимог, правил та принципів для визначення оцінки і відображення у фінансових звітах окремих суб'єктів бухгалтерського обліку.

Принципи бухгалтерського обліку — правило, яким слід керуватися при вимірюванні, оцінці та реєстрації господарських операцій і при відображенні їх результатів у фінансовій звітності.

Бухгалтерський облік та фінансова звітність ґрунтуються на таких основних принципах:

* обачність — застосування в бухгалтерському обліку методів оцінки, які повинні запобігати заниженню оцінки зобов'язань та витрат і завищенню оцінки активів і доходів підприємства;
* повне висвітлення - фінансова звітність повинна містити всю інформацію про фактичні та потенційні наслідки господарських операцій та подій, здатних вплинути на рішення, що приймаються на її основі;
* автономність — кожне підприємство розглядається як юридична особа, відокремлена від її власників, у зв'язку з чим особисте майно та зобов'язання власників не повинні відображатися у фінансовій звітності підприємства;
* послідовність — постійне (із року в рік) застосування підприємством обраної облікової політики. Зміна облікової політики можлива лише у випадках, передбачених національними положеннями (стандартами) бухгалтерського обліку, і повинна бути обґрунтована та розкрита у фінансовій звітності;
* безперервність — оцінка активів та зобов'язань підприємства здійснюється виходячи з припущення, що його діяльність буде тривати далі;
* нарахування та відповідність доходів і витрат - для визначення фінансового результату звітного періоду необхідно порівняти доходи звітного періоду з витратами, що були здійснені для отримання цих доходів. При цьому доходи і витрати відображаються в бухгалтерському обліку та фінансовій звітності в момент їх виникнення, незалежно від дати надходження або сплати грошових коштів;
* превалювання сутності над формою – операції обліковуються відповідно до їх сутності, а не лише виходячи з юридичної форми;
* історична (фактична) собівартість – пріоритетною є оцінка активів підприємства, виходячи з витрат на їх виробництво та придбання;
* єдиний грошовий вимірник — вимірювання та узагальнення всіх господарських операцій підприємства у його фінансовій звітності здійснюються в єдиній грошовій одиниці;
* періодичність — можливість розподілу діяльності підприємства на певні періоди часу з метою складання фінансової звітності.

Аналітичний облік (грец. analitykos — відповідний) — система бухгалтерських записів, що дає детальні відомості про рух господарських засобів; призначається для оперативного керівництва господарством і складання звітності; будується по кожному синтетичному рахунку окремо. Найбільш укрупнені і загальні для всіх підприємств галузі позиції аналітичного обліку передбачаються в плані рахунків і називаються субрахунками. У відмінність від синтетичного обліку, аналітичний облік ведеться не лише у вартісних, але і в натуральних показниках, а також містить довідкові дані. По синтетичних рахунках з найбільш розчленованою системою записів для аналітичного обліку застосовують окремі облікові регістри (картотеки, відомості і ін.) — для пооб'єктного обліку основних засобів по видах їх і місцям знаходження, складського кількісно-сортового обліку матеріалів і готової продукції, для особових рахунків розрахунків з робітниками і службовцями по заробітній платі, для обліку витрат в розрізі аналітичних позицій калькуляційних рахунків виробництва — по видах продукції, стадіях обробки, статтям калькуляції т.п. Записи аналітичного обліку по таких рахунках звіряють із записами синтетичного обліку за допомогою сальдових або оборотних відомостей, підсумки яких мають бути тотожні підсумкам записів у відповідному синтетичному рахунку. При менш розгалуженій номенклатурі аналітичних позицій — по збірно-розподільним рахункам, більшості розрахункових рахунків — записи аналітичного обліку поєднують в загальних регістрах із записами синтетичного обліку (накопичувальних відомостях, журналах-ордерах, табуляграмах і ін.).

Синтетичний облік – узагальнене віддзеркалення в грошовому вимірі економічно однорідних господарських засобів, їх джерел і господарських процесів. Ведеться в розрізі синтетичних рахунків бухгалтерського обліку, єдиний план яких затверджується в централізованому порядку. Синтетичний облік служить для здобуття сумарних відомостей про господарсько-фінансову діяльність підприємств, установ, організацій. Він має важливе значення: дає можливість перевіряти взаємозв'язок всіх господарських операцій і шляхом звірки синтетичного і аналітичного обліку контролювати повноту і правильність їх віддзеркалення в обліку. Матеріали синтетичного обліку, як зведеного обліку, використовуються для складання і перевірки звітності. На підприємстві за допомогою синтетичного обліку визначаються загальна наявність основних засобів сировини і матеріалів, витрати на виробництво, стан розрахунків з постачальниками і т. п. У фінансово-кредитних установах синтетичний облік використовується для обліку і контролю за виконання бюджету і кредитними операціями.

Перед складанням балансу в цілях звірки правильності записів на бухгалтерських рахунках і отримання узагальнених даних про рух господарських засобів, їх джерела і господарські процеси по аналітичних і синтетичних рахунках складаються оборотні відомості.

Будь-яка поважаюча себе компанія, в роботі якої присутні контрагенти і живі клієнти, повинна обов'язково вести їх докладний облік. Якщо говорити науковими словами, то таке явище називається впровадження CRM (в перекладі — управління взаємовідносинами клієнтів). Дана система орієнтована на різний бізнес з певним рівнем розвитку і масштабністю. Хтось обмежується адресною книгою в поштовому додатку, іншим за мільйони вигідно купувати глобальні клієнт-серверні розробки, що містять спеціалізовані системи обліку. Третім взагалі потрібно засіб для управління фінансами або прийняття різних рішень. У будь-якому випадку, якщо ви займаєтеся комерційною діяльністю, потрібно розташовувати програмним забезпеченням спеціального призначення. З його допомогою зберігаються відомості про контрагентів. Наприклад, це можуть бути виписані рахунки, дні народження і багато іншого. Представники малого і середнього бізнесу, які не знають що таке зверх-прибуток, зацікавлені в простому і недорогому рішенні.

* 1. **Огляд існуючих програмних рішень**

*Дебет Плюс* - українська кросплатформенна автоматизована система управління підприємством (АСУП) і програмна платформа, написана поверх Eclipse RCP. Володіє багатими можливостями конфігурування. В якості СКБД можна використовувати вбудовану Apache Derby, Open Source SQL-сервери бази даних PostgreSQL і MySQL або, для платної корпоративної версії, комерційні Oracle, Microsoft SQL Server або DB2.

Вся інформація в системі зберігається у вигляді первинних документів, проведень, довідників і початкових залишків. Система "Дебет Плюс" побудована за модульним принципом: для ведення бухгалтерського, управлінського, банківського, касового обліку, обліку заробітної плати і кадрів використовуються модулі (підсистеми), що автоматизують ведення відповідного розділу обліку. Стандартна базова конфігурація «Дебет Плюс v12» включає 11 підсистем:

* Зведення бухгалтерського балансу;
* Облік банківських операцій;
* Облік касових операцій;
* Облік товарно-матеріальних цінностей (склад, торгівля, виробництво);
* Облік основних засобів;
* Облік заробітної плати;
* Облік кадрів;
* Управлінський облік;
* CRM (Взаємодія з клієнтами);
* Автотранспорт (облік транспортних засобів, затрат ПММ, ведення шляхових листів);
* Адміністратор комплекса.

Можливе автономне використання окремої підсистеми, при цьому сама система працює як єдине ціле — проведення, зроблене в одній підсистемі, видно у всіх решта, не потрібно ніяких додаткових операцій по перенесенню інформації. Податковий облік як підсистема не виділяється, журнали податкового обліку знаходяться в підсистемі «Облік ТМЦ».

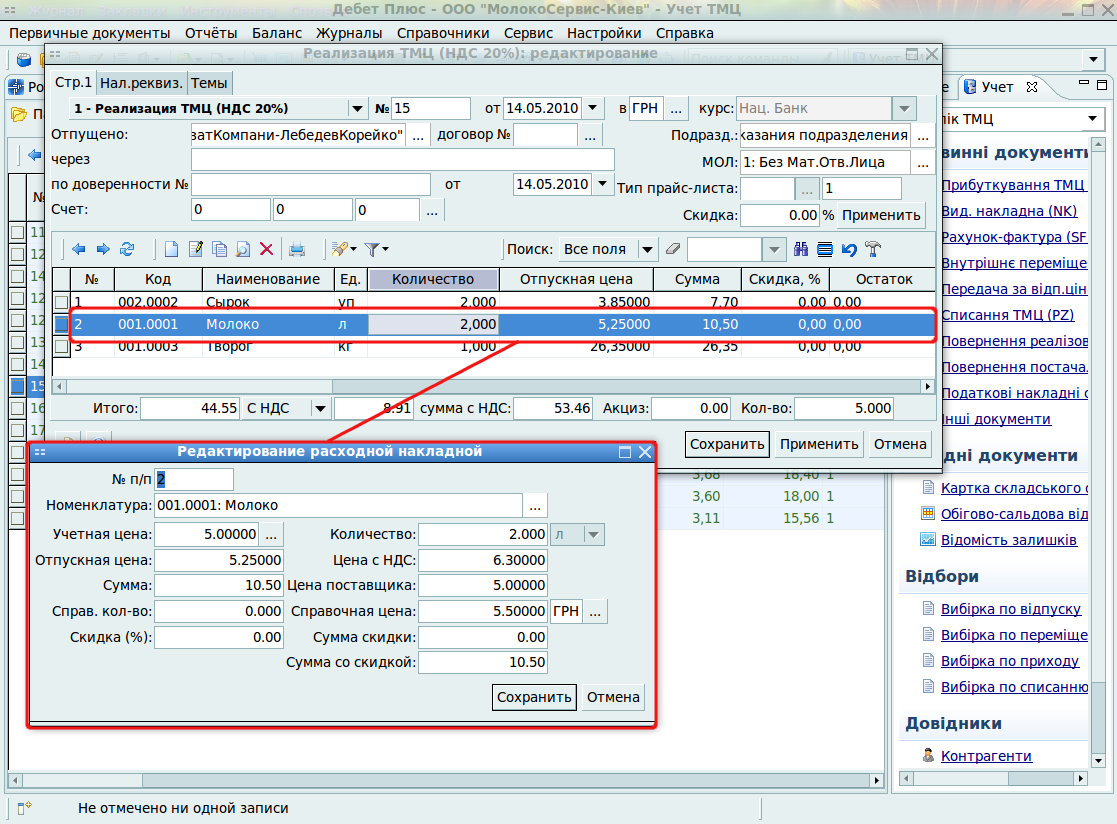


Рисунок 1.1 – Скріншот програми Дебет Плюс

*Облік клієнтів* - програма ведення бази даних та обліку клієнтів (організацій і осіб), виставлених їм рахунків, укладених договорів, інших документів, продуктів (товарів і послуг), співробітників.

Програма призначена для автоматизації діяльності відділу продаж, фінансового відділу та інших завдань, які можна гнучко налаштувати. Є різні варіанти контролю оплати рахунків. Зручне сортування та фільтрування по будь-якому полю, експорт безпосередньо в Excel, генерація документів Word на основі шаблонів (договорів, рахунків), мережний і багатокористувацький режими, ведення історії змін. Ви можете створювати нові таблиці в базі даних, додавати нові поля до них (як збережені, так і обчислювані, у тому числі за формулами типу SELECT...), зберігати в базі даних картинки, визначати спадаючі списки і значення в них, прив'язувати підлеглі таблиці, вважати підсумки за формулами, будувати дерево по заданих полях та інші зручності.

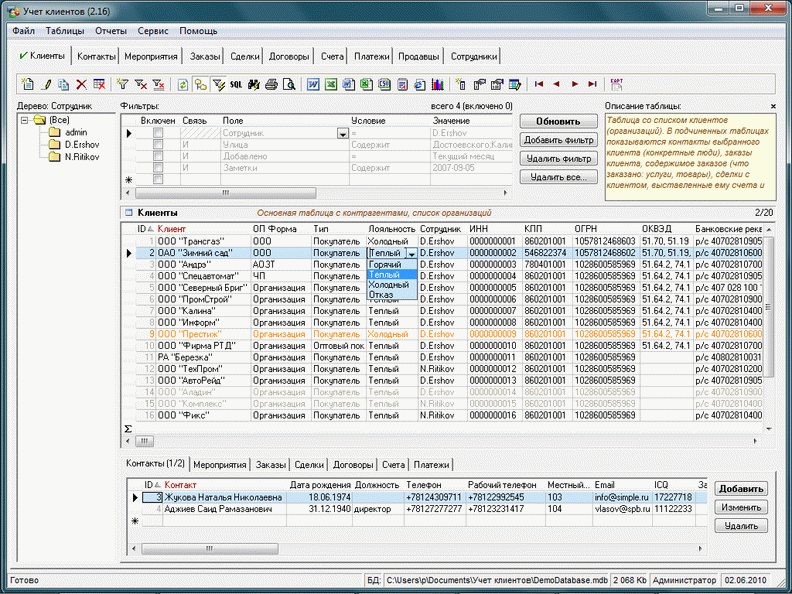


Рисунок 1.2 – Скріншот програми Облік клієнтів

Платформа *Supasoft* - універсальна програма для обліку ваших даних. Таких як клієнти, продажа, замовлення, партнери, постачальники, договори, відвантаження, оплати, доходи, витрати, фінанси, товари, продукти, послуги, техніка, матеріали, комплектуючі тощо. За будь-якими даними, які у вас є, можна налаштувати облік. Ви отримаєте програму, в якій будуть тільки потрібні вам дані і не буде нічого зайвого. За будь-яких полях, які ви додали можна проводити пошук, обмежувати доступ користувачам, будувати звіти і багато іншого. При великій кількості користувачів, різним користувачам можна дозволяти або забороняти бачити певні поля, стовпці, рядки, функції, пункти меню і т.п. Конструктор звітів дозволяє створювати самі різні звіти за всіма даними, які є у вашій базі.

За допомогою платформи Supasoft ви отримаєте рішення на 100% заточене під ваші вимоги.

Основні можливості:

* Налаштування структури даних . Розробка вашої конфігурації з нуля.
* Додавання власних полів . У програмі будуть тільки потрібні вам поля.
* Створення формул для полів . Наприклад, для обчислення ПДВ, знижки, прибутку, зарплати і т.п.
* Налаштування карток . Розташуйте поля на картці як вам подобається.
* Налаштування таблиць . Включіть в таблицях тільки потрібні вам стовпці.
* Фільтри . Пошук даних у таблицях за будь-яких полях.
* Права доступу . Права доступу до даних по користувачам і групам користувачів: читання, зміна, видалення.
* Шаблони документів . Такі як рахунок, договір, акт, накладна, кошторис і т.п.
* Звіти . Майже як зведені таблиці в Excel. З будь-яких даних у базі.
* Імпорт даних . Будь-які дані, які у вас є, можна імпортувати в створену вами базу.
* Експорт даних . Будь-які дані можна експортувати.
* Нагадування . За будь-яких полях типу "Дата і час", можна налаштувати нагадування.
* Багатовалютний облік . Якщо у вашій роботі використовується кілька валют, то це можна налаштувати.

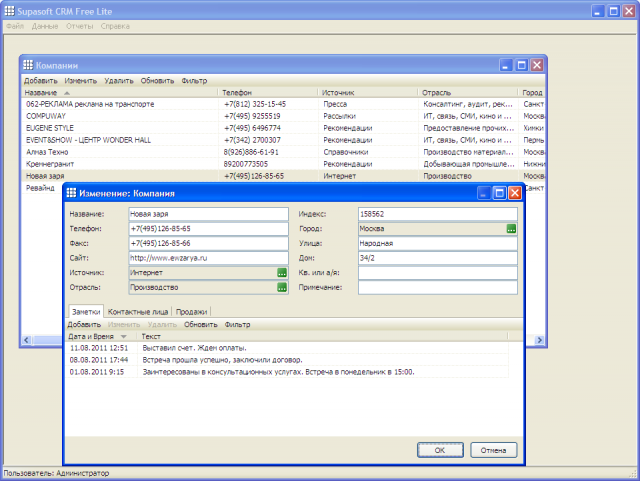


Рисунок 1.3 – Скріншот програми Supasoft

*1С: Підприємство 8* – програмний продукт ”1С:Бухгалтерія 8 для України” включає технологічну платформу “1С:Підприємство 8” і конфігурацію (прикладне рішення) “Бухгалтерія для України”.

Правила ведення обліку налаштовані саме в конфігурації. “1С:Бухгалтерія 8 для України” призначена для автоматизації бухгалтерського й податкового обліку, включаючи підготовку обов'язкової (регламентованої) звітності, в організаціях, що здійснюють які-небудь види комерційної діяльності: оптову й роздрібну торгівлю, комісійну торгівлю (включаючи субкомісію), надання послуг, виробництво і т.д. Бухгалтерський і податковий облік ведеться відповідно до чинного законодавства України. “1С:Бухгалтерія 8 для України” забезпечує рішення всіх завдань, які виникають перед бухгалтерською службою підприємства, якщо бухгалтерська служба повністю відповідає за облік на підприємстві, включаючи, наприклад, виписку первинних документів, облік продажів і т.д. Дане прикладне рішення також можна використовувати винятково для ведення бухгалтерського й податкового обліку, а завдання автоматизації інших служб, наприклад, відділу продажів, вирішувати спеціалізованими конфігураціями або іншими системами.

Пакет “1С: Підприємство” складається з таких модулів:

1. Версія “1С: Підприємство 7.7”:

- 1С: Бухгалтерія;

- 1С: Торгівля і Склад;

- 1С: Зарплата і кадри.

2. Версія “1С: Підприємство 8.0”:

- 1С: Бухгалтерія 8.0;

- 1С: Управління торгівлею 8.0;

- 1С: Зарплата і управління персоналом 8.0.

До складу “1С:Бухгалтерії 8 для України” включений план рахунків бухгалтерського обліку, відповідний до Наказу Міністерства Фінансів України “Про затвердження Плану рахунків бухгалтерського обліку та Інструкції про його використання” від 30.11.99р. №291. Склад рахунків, організація аналітичного, валютного, кількісного обліку на рахунках відповідають вимогам законодавства по веденню бухгалтерського обліку й відбиттю даних у звітності. При необхідності користувачі можуть самостійно створювати додаткові субрахунки й розрізи аналітичного обліку.

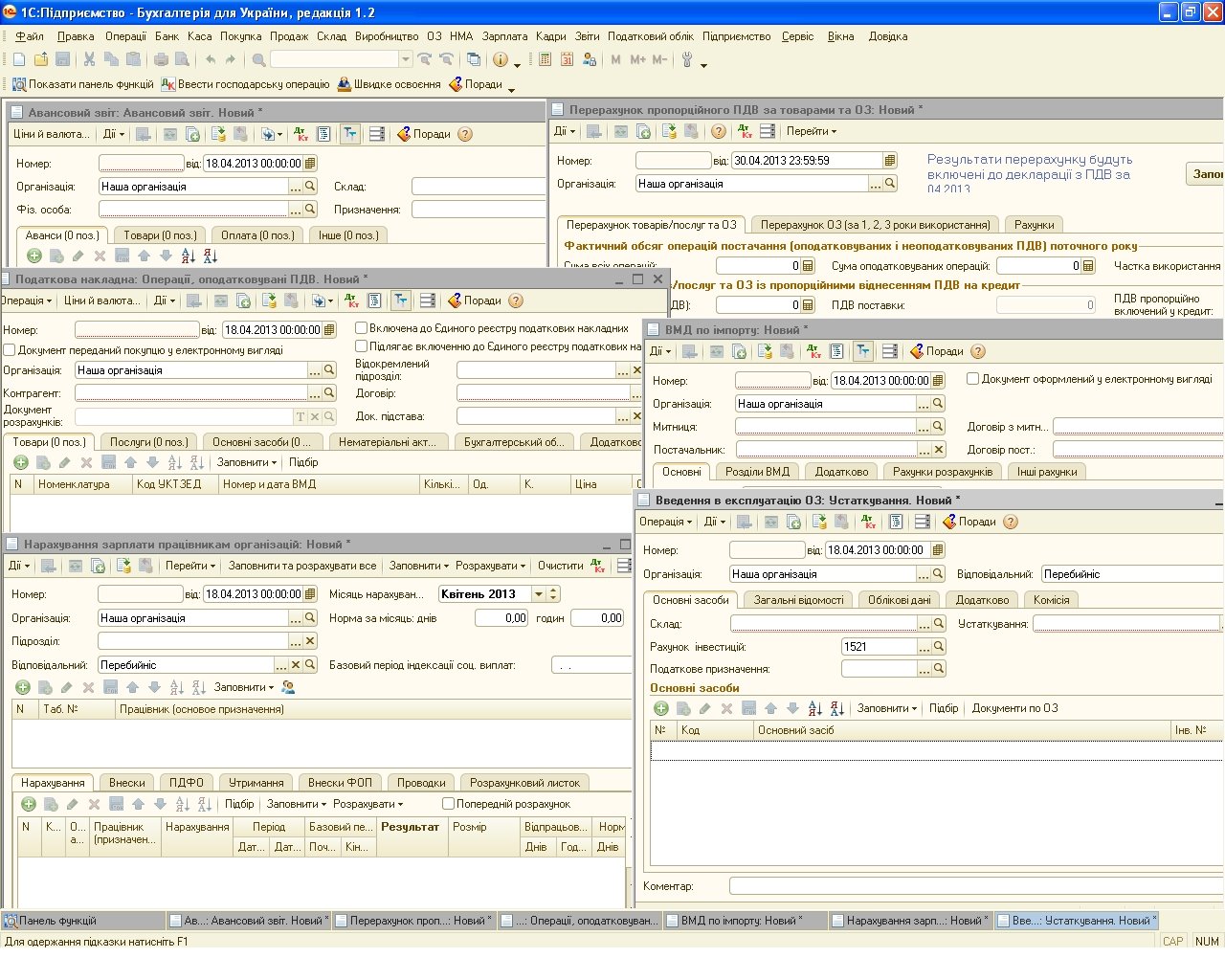


Рисунок 1.4 – Скріншот програми 1С:Бухгалтерія

*Акцент-бухгалтерія* – це універсальна автоматизована програма для ведення бухгалтерського, управлінського та фінансового обліку на підприємствах. Ця програма забезпечує повноцінний правильний з бухгалтерської точки зору багатовимірний та багаторівневий аналітичний і синтетичний облік. В програмі реалізована загальноприйнята методика бухгалтерського фінансового та податкового обліку, яка підтримує НСБО. “Стандартне рішення” що входить в комплект поставки продукту, створене для підприємств оптово-роздрібної торгівлі. Також напрацьовано ряд спеціалізованих рішень, які базуються на стандартах і враховують специфіку бухгалтерського обліку при автоматизації підприємств.

“Акцент-бухгалтерія” версій 6.0 та 7.0 включає:

1. Нарахування та виплата зарплати.

2. Автоматизований розрахунок різних видів нарахувань і стягнень.

3. Облік необоротних активів та балансової вартості основних засобів.

4. Облік валових доходів та витрат.

5. Розрахунки з бюджетом.

6. Розрахунки з підзвітними особами.

7. Розрахунки з постачальниками та покупцями.

8. Облік банківських та касових операцій.

9. Можливість формування, зберігання та друку всіх необхідних документів.

*Парус-Підприємство* – повнофункціональна система, що дозволяє автоматизувати всі облікові й управлінські завдання, характерні для підприємств малого і середнього бізнесу. У її складі тепер десять модулів:

– “Парус-Адміністратор”;

– “Парус-Бухгалтерія”;

– “Парус-Реалізація і Склад”;

– “Парус-Заробітна плата”;

– “Парус-Персонал”;

– “Парус-Менеджмент і Маркетинг” (CRM-модуль);

– “Парус-Консолідація”;

– “Парус-Ресторан”;

– “Парус-Готель”;

– “Парус-Лікувальна установа”.

*Бест звіт плюс* – програмний комплекс призначений для автоматизації процесів роботи зі звітною документацією встановленого зразка. Він забезпечує організацію електронного документообігу у всіх без винятку суб'єктів господарювання будь-якої форми власності та джерел фінансування або між ними та державними контролюючими органами, подача звітності яким передбачена чинним законодавством. Це можливість інтеграції даних з будь-якої бухгалтерської програми (1C, і т.д.) Передача звітності засобами електронної пошти із застосуванням підсистеми шифрування та пересилки.

Функціональні можливості програмного забезпечення:

1. Формування картки установи.

2. Формування та ведення реєстру форм (бланків) звітних документів.

3. Формування звітних документів із використанням різноманітних функцій контролю

4. Імпорт інформації зі встановлених на підприємстві автоматизованих систем бухгалтерського обліку та управління виробництвом.

5. Формування пакетів звітності в електронному вигляді електронною поштою із застосуванням криптографічного захисту.

Отже, кожне підприємство може вибрати програмний продукт для автоматизації обліку відповідно до своїх особливостей ведення діяльності та особливостей ведення бухгалтерського обліку.

**2 ПРОЕКТУВАННЯ СТРУКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА АЛГОРИТМІВ ФУНКЦІОНУВАННЯ ПРОГРАМИ**

* 1. **Вибір програмних засобів для реалізації проекту**

Значна кількість сучасних мов програмування має розвинуті візуальні засоби для розробки графічного інтерфейсу, причому здійснюється програмування розміщених на спеціальних формах об'єктів з настроюванням їх властивостей та поведінки. CodeGear Delphi і C++ Builder, Microsoft Visual Studio та мови, які включає в себе цей засіб (Visual Basic, Visual C#, Visual J# тощо) часто плутають з візуальними мовами програмування. Всі ці мови є текстовими, а не візуальними (графічними). MS Visual Studio та Delphi є візуальними середовищами програмування.

Замовником програмного продукту було вибрано середовище програмування Delphi.

Embarcadero Delphi — це інтегроване середовище швидкої розробки програмного забезпечення для роботи під Microsoft Windows. Воно підтримує розробку Windows-застосунків на мові програмування Delphi, яка є наступницею мови Object Pascal.

Delphi в основному використовується для розробки настільних застосунків та корпоративних СКБД, проте цей інструмент можна використовувати для розробки будь-якого загального програмного забезпечення. Не залишена осторонь і можливість побудови веб-застосунків.

* + 1. *Delphi*

Turbo Pascal та Borland Pascal були дешевими 16-бітними компіляторами. За роки свого існування, вони пройшли через багато релізів, і в основному використовувалися для створення програм, що виводили інформацію у текстовому режимі. Коли використання графічного інтерфейсу користувача стало необхідним у Microsoft Windows 3.1, було представлено Delphi, розроблене на основі Borland Pascal. Delphi була першою так званою системою швидкої розробки, випущеною у 1995-ому році для 16-бітної Windows 3.1.

Delphi 2, представлена роком пізніше, підтримувала 32-бітне Windows-середовище, a версія, що використовувала мову C++, під назвою C++ Builder побачила світ іще кількома роками пізніше.

Головним архітектором Delphi на той час був Андерс Гейлсберг, який розробив Turbo Pascal. Він перейшов у Microsoft у 1996 для розробки мови C#.

У 2001-ому році була представлена версія для операційної системи Linux під назвою Kylix. Проте дуже швидко вона була розкритикована за низьку якість і велику кількість помилок. Через це, враховуючи низький рівень продажу, вона була занедбана після третьої версії.

Була зроблена спроба зробити підтримку і Linux, і Windows для багатоплатформної розробки, внаслідок чого у Delphi 6 була включена бібліотека CLX, багатоплатформна версія бібліотеки VCL. Технологія CLX теж зазнала поразки і після падіння Kylix теж була закинута.

Ще починаючи з першої версії 1.0 розробка програм для баз даних стала однією з сильних сторін Delphi. Бібліотека візуальних компонент (англ. Visual Component Library, VCL) містила велику бібліотеку компонент для доступу та маніпулювання з базами даних. Borland Database Engine була оригінальною технологією зв'язку з базами даних, і була єдиним рішенням у ранніх версіях Delphi. Навіть зараз, коли вона рідко використовується, остання версія IDE все ще поставляється з BDE, необов'язковим для встановлення.

Delphi 7, випущена у серпні 2002, стала стандартом де-факто для багатьох Delphi-розробників, і навіть зараз вона активно використовується. В Delphi 7 додано підтримку для тем Windows XP і покращено можливості для побудови Web-застосунків. Також це була остання версія Delphi, яка могла використовуватися без активації. Вона мала лише необов'язкову реєстрацію, яку можна було просто проігнорувати. Delphi 7 є найбільш оціненою IDE, створеною Borland завдяки своїй стабільності, швидкості і низькими вимогами до апаратного забезпечення. Попри це у цій версії Delphi, як і у всіх інших, була велика кількість відомих помилок, так і ніколи не виправлених Borland. Завдання виправлення цих помилок компанія залишила на спільноту Delphi (дивіться «проект відродження Delphi»).

Delphi 8, представлений у грудні 2003-ого, був лише .NET-релізом, що дозволяв розробникам компілювати вихідні коди Object Pascal у .NET CIL. Також він дуже відрізнявся від попередників зовнішнім виглядом IDE, в якому вперше застосовано багатовіконний стиль, багато в чому схожий на середовище Microsoft's Visual Studio.NET. Хоча можливість перемикання у класичне (англ. Classic Undocked) розміщення вікон все ще залишилася.

Наступна версія Delphi 2005 (Delphi 9) включала в собі можливість розробки як для платформи Microsoft .NET, так і під «рідну» Win32. Також ця версія дозволяла маніпулювати даними з баз даних ще у режимі дизайнера. Відзначалося і вдосконалене IDE, а з мовних можливостей був новий вираз for … in (аналог foreach у C#'і). Проте, середовище було знову розкритиковане за виявлені помилки — обидві версії Delphi 8 і Delphi 2005 мали проблеми із стабільністю, які були лише частково виправлені у сервісних пакетах.

Наприкінці 2005-ого було випущено Borland Developer Studio 2006, яка включала C#, Delphi.NET, Delphi Win32 та C++ у єдиному IDE. Ця версія була набагато стабільніша за Delphi 8 чи Delphi 2005, і ще більше поліпшена з випуском сервісних пакетів та оновлень.

Восьмого лютого 2006-ого року Borland оголосила про пошук покупця для її IDE і лінії продуктів для баз даних, що включало Delphi. Це рішення компанія пояснила намаганням сконцентруватися на своїй лінії продуктів ALM. Ця новина була сприйнята з неоднозначною реакцією розробників, які все ще не полишили Delphi.

Шостого вересня 2006-ого Developer Tools Group (робоча назва ще не відділеної групи розробників) компанії Borland Software Corporation випустили одномовні версії Borland Developer Studio, повернувшись до популярного імені Turbo. Набір «Turbo» продуктів включав Turbo Delphi for Win32, Turbo Delphi for .NET, Turbo C++, та Turbo C#. Кожна версія доступна в двох редакціях: Explorer — версія для вільного скачування; та Professional — дешева (US$899 для нових користувачів, US$399 для оновлення) версія, яка відкривала доступ до тисяч компонент сторонніх виробників. На відміну від ранніх Personal версій Delphi, нові Explorer-редакції могли використовуватися для комерційної розробки програмного забезпечення.

14 листопада 2006 Borland оголосила про відділення групи розробників у незалежну дочірню компанію CodeGear.

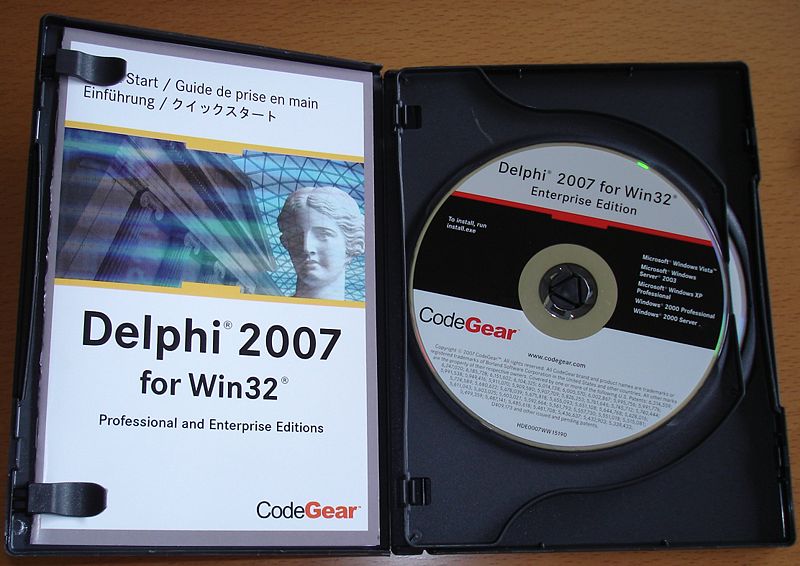


Рисунок 2.1 – Фото коробки Delphi 2007, випущеного CodeGear

Delphi 2007 — перша версія випущена CodeGear 16-ого березня 2007. Win32-версія була представлена першою, пізніше була випущена .NET-версія Delphi 2007 як частина продукту CodeGear RAD Studio 2007. Нові можливості включали підтримку MS Build та вдосконалення Visual Component Library для Windows Vista. CodeGear також представила DBX4 як нову версію dbExpress. Вперше Delphi можна було завантажити з інтернету і активувати ліцензійним ключем. Локалізовані версії Delphi 2007 були одночасно представлені на англійській, французькій, німецькій та японській мовах. RAD Studio 2007, яка включала розробку на Delphi.NET та C++, була випущена 5-ого вересня 2007-ого.

У Delphi 2009 (кодова назва Tiburón) додано багато нових можливостей, зокрема узагальнення (англ. generics), анонімні методи (для Win32 та .NET), повністю перероблено VCL та RTL для повної підтримки Юнікоду.

Borland продала CodeGear компанії Embarcadero Technologies в 2008. Embarcadero зберегла відділ CodeGear, створений Borland, для ідентифікації куплених продуктів, свої ж розробки Embarcadero вирішила розповсюджувати під іменем DatabaseGear.

25 серпня 2009 року було випущено 14-ту (13-ту версію розробники пропустили) версію — Delphi 2010. Її було наділено такими новинками:

* Підтримка API Windows 7, у тому числі Direct2D
* Підтримка жестів у Windows 2000, XP, Vista та 7. У Windows 2000 та XP використовується внутрішній жестовий двигун
* IDE Insight - пошук по IDE, що надає доступ до будь-якої її функції або параметру
* Розширений RTTI, підтримка атрибутів
* Класичний інтерфейс Delphi 7

Під час бета-тестування наступної версії делфі - Delphi XE, у ній була присутня можливість створення програм для Mac OS X. Це було можливо завдяки бібліотеці UCL що була по суті оновленою CLX і базувалася на Qt 4. В UCL було дуже багато багів, які було важко знаходити та виправляти через дику суміш delphi та c++-коду. В решті-решт розробники вирішили, що обраний підхід не дозволить їм створити якісний продукт, і 30 серпня 2010 року 15-ту версію Delphi було випущено без підтримки Mac OS X. Досить бідна на новинки версія отримала інтеграцію з Subversion, чисельні виправлення та дрібні покращення, а також солідну кількість корисних утиліт від сторонніх розробників у вигляді безкоштовного додатка.

2 вересня 2011 випущено 16-ту версію Delphi XE2. Серед головних нововведень версії:

* Підтримка 64-бітної архітектури під Windows
* Можливість створення застосунків для Mac OS X та iOS. Для цих платформ є нова бібліотека компонентів — FireMonkey (у минулому DXScene від KsDev). FireMonkey повністю написана на Delphi, використовує векторну графіку й апаратне прискорення графічного процесора
* З'єднувачі DataSnap для мобільних пристроїв
* Система зв'язування даних LiveBindings
* У бібліотеку VCL було додано підтримку графічних стилів оформлення.

3 вересня 2012 на світ з'явилася сімнадцята версія делфі — Delphi XE3. В цій версії бібліотека FireMonkey отримала подальший розвиток. У неї додано підтримку растрових стилів, якорів (Anchors), дій (Actions), відео та аудіо, а також проведено цілий ряд оптимізацій та рефакторингів. Система зв'язування даних LiveBindings отримала графічний редактор, що цілком відповідає концепції RAD програмування, якого з перших версій дотримується Delphi. Для VCL та FireMonkey були створені стилі Metropolis, що імітують оформлення WinRT програм для Windows 8. Створення повноцінних WinRT програм на делфі на даний час неможливе через неможливість створення стороннього нативного компілятора Third-party compilers locked out of Windows Runtime development. Підтримка iOS була виключена з XE3, вона з'явиться на початку 2013-го року у вигляді окремого продукту — Mobile Studio.

* + 1. *Переваги мови програмування*

Простота, швидкість і ефективність Delphi пояснюють її популярність. Delphi має один з швидких компіляторів, що породжує, тим не менш, дуже і дуже непоганий об'єктний код. Є й інші переваги: простота вивчення Object Pascal; полегшують життя властивості (properties); програми, написані на Delphi, не потрібно постачати додатковими бібліотеками (на відміну від зв'язки C + +/MFC). VCL надає зручний, легко розширюваний об'єктно-орієнтований інтерфейс до Windows, що ні в якій мірі не заважає програмісту опускатися в глибини Windows API. Як було сказано вище, модель програмування в Delphi - компонентна, що дозволяє користуватися компонентами, написаними іншими розробниками, навіть не маючи їх вихідного коду. В Інтернеті є величезна кількість компонентів, значна частина яких розповсюджується безкоштовно. Застосування компонентної моделі призводить до того, що досить багато чого в поведінці об'єктів програмувати не потрібно взагалі, і багато, на що в інших середовищах пішли б тижні, можна зробити за години або навіть хвилини. Воно і зрозуміло - адже це RAD-середовище. До достоїнств можна віднести дуже швидкий браузер класів і миттєвий висновок підказки автозавершення коду (code completion). Якщо коротко - може все. Звичайно, Object Pascal накладає певні обмеження, але для тих речей, для яких її писали, Delphi підходить практично оптимально.

* + 1. *Недоліки мови програмування*

Їх мало, але вони є. Головний, на мій погляд, недолік (і одночасно гідність) - статичне приєднання (linking) бібліотеки VCL і компонентів до виконуваного файлу. Треба сказати, що VCL можна приєднувати статично і динамічно, але тоді з кожним своїм додатком вам доведеться поширювати ще й VCL, а це більше 5 Мбайт. Однак якщо не захоплюватися інтерфейсними "наворотами" і використовувати в програмі мінімально необхідне число компонентів, то виконуваний файл буде невеликий. Інший недолік (і знову ж гідність) полягає в тому, що в використовуваної в Delphi парадигмі форм (Forms) вся інформація про форму, включаючи властивості, настройки компонентів, значення за замовчуванням, зберігається в exe-файлі, причому не оптимальним чином. Аналіз вихідного коду VCL показує, що при створенні форми фактично відбувається мало не синтаксичний розбір даних ініціалізації, що не може її не сповільнювати. Третій недолік, який хто-небудь теж може назвати гідністю, - це Object Pascal. Незважаючи на простоту, ефективність і легкість у вивченні, йому не вистачає дуже багатьох потужних засобів інших мов програмування.

* 1. **Огляд СКБД MySQL та її застосування в Delphi для розробки програмного продукту**
     1. *Версії MySQL*

MySQL — вільна система керування реляційними базами даних.

MySQL був розроблений компанією «ТсХ» для підвищення швидкодії обробки великих баз даних. Ця система керування базами даних (СКБД) з відкритим кодом була створена як альтернатива комерційним системам. MySQL з самого початку була дуже схожою на mSQL, проте з часом вона все розширювалася і зараз MySQL — одна з найпоширеніших систем керування базами даних. Вона використовується, в першу чергу, для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування.

MySQL виникла як спроба застосувати mSQL до власних розробок компанії: таблиць, для яких використовувалися ISAM — підпрограми низького рівня. У результаті був вироблений новий SQL-інтерфейс, але API-інтерфейс залишився в спадок від mSQL. Звідки походить назва «MySQL» — достеменно не відомо. Розробники дають два варіанти: або тому, що практично всі напрацювання компанії починалися з префікса My, або на честь дівчинки на ім'я My, дочки Майкла Монті Віденіуса, одного з розробників системи.

Логотип MySQL у вигляді дельфіна носить ім'я «Sakila». Він був обраний з великого списку запропонованих користувачами «імен дельфіна». Ім'я «Sakila» було відправлено Open Source-розробником Ambrose Twebaze

В січні-лютому 2008 Sun Microsystems придбала розробника системи керування базами данних MySQL за $1 млрд. Після поглинання у 2009 році Sun Microsystems компанією Oracle Corporation MySQL стала власністю Oracle.

Версія MySQL 5.1 продовжує шлях до стандарту SQL:2003. MySQL 5.1 містить такі нововведення:

* Сегментування — можливість розбити одну велику таблицю на декілька частин, розміщених в різних файлових системах, базуючись на визначеній користувачем функції. При деяких умовах це може дати серйозне збільшення продуктивності та, крім того, полегшує масштабування таблиць.
* Змінено поведінку ряду операторів для забезпечення більшої сумісності зі стандартом SQL:2003.
* Порядкова реплікація (row-based реплікація), при якій в бінарний лог буде записуватись тільки інформація про реально змінені рядки таблиці замість оригінального (і, можливо, більш повільного) тексту запиту.
* Вбудований планувальник робіт, що періодично запускаються. По синтаксису додання задачі схоже на додання тригера до таблиці; по ідеології — на crontab.
* Додатковий набір функцій для обробки XML, реалізація підтримки XPath.
* Нові засоби діагностики проблем і утиліти для аналізу продуктивності. Розширено можливості з керування вмістом лог-файлів, логи тепер можуть бути збережені і в таблицях general\_log і в slow\_log. Утиліта mysqlslap дозволяє провести тестування навантаження БД із записом часу реакції на кожний запит.
* Для спрощення операції оновлення підготовлена утиліта mysql\_upgrade, яка виконає перевірку всіх існуючих таблиць на предмет сумісності з новою версією, і при необхідності виконає належні коригування.
* MySQL Cluster тепер йде як окремий продукт, який базується на MySQL 5.1 і сховищі NDBCLUSTER.
* Значні зміни в роботі MySQL Cluster, такі, як, наприклад, можливість зберігання табличних даних на диску.
* Повернення до використання вбудованої бібліотеки libmysqld, відсутньої в MySQL 5.0.
* API для плагінів, що дозволяє завантажувати сторонні модулі, які розширюють функціональність (наприклад, повнотекстовий пошук), без перезапуску сервера.
* Реалізація парсера повнотекстового пошуку у вигляді plug-in.
* Новий рушій таблиць Maria (стійкий до збоїв клон MyISAM)[5], який у 2010 був перейменований на Aria та став основою форку MySQL від Монті Віденіуса під назвою MariaDB.

Тип таблиць Maria.

Maria (починаючи з версії 5.2.x — Aria) — новий тип таблиць MySQL для зберігання даних. Maria являє собою розширену версію сховища MyISAM, з додаванням засобів збереження цілісності даних після краху.

Основні переваги Maria:

* У разі краху проводиться відкат результатів виконання поточної операції або повернення в стан до команди LOCK TABLES. Реалізація через ведення логу операцій.
* Можливість відновлення стану з будь-якої точки в журналі операцій, включаючи підтримку CREATE/DROP/RENAME/TRUNCATE. Може бути використана для створення інкрементальних бекапів, через періодичне копіювання лог файлу.
* Підтримка всіх форматів стовпців MyISAM, розширена новим форматом «rows-in-block», що використовують сторінковий спосіб зберігання даних, при якому дані в стовпцях можуть кешуватися.
* В майбутньому буде реалізовано два режими: транзакційний і без відображення в журналі транзакцій, для не критичних даних.
* Розмір сторінки даних дорівнює 8Кб (в MyISAM 1Кб), що дозволяє досягти більш високої продуктивності для індексів за полями фіксованого розміру, але працює повільніше у разі індексування ключів змінної довжини.

Гілка MySQL 5.5 містить ряд значних поліпшень, пов'язаних з підвищенням масштабованості та швидкодії, серед яких:

* Використання за замовчуванням рушія InnoDB.
* Підтримка напівсинхронного (semi-synchronous) механізму реплікації, заснованого на патчах до InnoDB від компанії Google.
* Поліпшення функцій з партіціювання даних. Розширений синтаксис для розбиття великих таблиць на кілька частин, розміщених в різних файлових системах (partitioning). Додані операції RANGE, LIST і метод оптимізації «partition pruning».
* Новий механізм оптимізації вкладених запитів та операцій JOIN.
* Перероблена система внутрішніх блокувань.
* Інтегровані патчі Google з оптимізацією роботи InnoDB на CPU з великою кількістю ядер.

MySQL — компактний багатонитевий сервер баз даних. Характеризується великою швидкістю, стійкістю і простотою використання.

MySQL вважається гарним рішенням для малих і середніх застосувань. Сирцеві коди сервера компілюються на багатьох платформах. Найповніше можливості сервера виявляються в UNIX-системах, де є підтримка багатонитковості, що підвищує продуктивність системи в цілому.

Можливості сервера MySQL:

* простота у встановленні та використанні;
* підтримується необмежена кількість користувачів, що одночасно працюють із БД;
* кількість рядків у таблицях може досягати 50 млн;
* висока швидкість виконання команд;
* наявність простої і ефективної системи безпеки.
  + 1. *Синтаксис операторів*

Оператор SELECT має наступну структуру:

SELECT [STRAIGHT\_JOIN]

[SQL\_SMALL\_RESULT][SQL\_BIG\_RESULT][SQL\_BUFFER\_RESULT]

[SQL\_CACHE | SQL\_NO\_CACHE] [SQL\_CALC\_FOUND\_ROWS] [HIGH\_PRIORITY]

[DISTINCT | DISTINCTROW | ALL]

select\_expression, ...

[INTO {OUTFILE | DUMPFILE} 'file\_name' export\_options]

[FROM table\_references

[WHERE where\_definition]

[GROUP BY {unsigned\_integer | col\_name | formula} [ASC | DESC], ...]

[HAVING where\_definition]

[ORDER BY {unsigned\_integer | col\_name | formula} [ASC | DESC], ...]

[LIMIT [offset,] rows]

[PROCEDURE procedure\_name]

[FOR UPDATE | LOCK IN SHARE MODE]]

SELECT застосовується для вилучення рядків, вибраних з однієї або декількох таблиць. Вираз select\_expression задає стовпці, в яких необхідно проводити вибірку. Крім того, оператор SELECT можна використовувати для витягання рядків, обчислених без посилання якусь таблицю. Наприклад:

mysql> SELECT 1 + 1;

- > 2

При вказівці ключових слів слід точно дотримуватися порядку, вказаний вище. Наприклад, вираз HAVING повинне розташовуватися після всіх виразів GROUP BY і перед усіма виразами ORDER BY .

Використовуючи ключове слово AS , висловом у SELECT можна призначити назву. Псевдонім використовується в якості імені стовпця в даному вираженні і може застосовуватися в ORDER BY або HAVING . Наприклад:

mysql> SELECT CONCAT (last\_name, ',', first\_name) AS full\_name

FROM mytable ORDER BY full\_name;

Псевдоніми стовпців не можна використовувати у виразі WHERE , оскільки знаходяться в стовпцях величини на момент виконання WHERE можуть бути ще не визначені.

Вираз FROM table\_references задає таблиці, з яких належить витягувати рядка. Якщо вказано ім'я більш ніж однієї таблиці, слід виконати об'єднання.

table\_name [[AS] alias] [USE INDEX (key\_list)] [IGNORE INDEX (key\_list)]

Оператор SELECT може бути представлений у формі SELECT ... INTO OUTFILE 'file\_name' . Цей різновид команди здійснює запис вибраних рядків у файл, зазначений у file\_name . Цей файл створюється на сервері і до цього не повинен існувати (таким чином, крім іншого, запобігається руйнування таблиць і файлів, таких як `/ etc / passwd ' ). Для використання цієї форми команди SELECT необхідні привілеї FILE . Форма SELECT ... INTO OUTFILE головним чином призначена для виконання дуже швидкого дампа таблиці на серверному комп'ютері. Команду SELECT ... INTO OUTFILE не можна застосовувати, якщо необхідно створити результуючий файл на іншому хості, відмінному від серверного. У такому випадку для генерації потрібного файлу замість цієї команди слід використовувати деяку клієнтську програму на зразок mysqldump - tab або mysql-e "SELECT ..." > Outfile . Команда SELECT ... INTO OUTFIL E є додатковою по відношенню до LOAD DATA INFIL E;

MySQL підтримує наступний синтаксис оператора JOIN при використанні в командах SELECT :

table\_reference, table\_reference

table\_reference [CROSS] JOIN table\_reference

table\_reference INNER JOIN table\_reference join\_condition

table\_reference STRAIGHT\_JOIN table\_reference

table\_reference LEFT [OUTER] JOIN table\_reference join\_condition

table\_reference LEFT [OUTER] JOIN table\_reference

table\_reference NATURAL [LEFT [OUTER]] JOIN table\_reference

{Oj table\_reference LEFT OUTER JOIN table\_reference ON conditional\_expr}

table\_reference RIGHT [OUTER] JOIN table\_reference join\_condition

table\_reference RIGHT [OUTER] JOIN table\_reference

table\_reference NATURAL [RIGHT [OUTER]] JOIN table\_reference

де table\_reference визначено, як:

table\_name [[AS] alias] [USE INDEX (key\_list)] [IGNORE INDEX (key\_list)]

і join\_condition визначено, як:

ON conditional\_expr |

USING (column\_list)

Ніколи не слід вказувати в частині ON які б то не було умови, що накладають обмеження на рядки в наборі результатів. Якщо необхідно вказати, які рядки повинні бути присутніми в результаті, слід зробити це у вираженні WHERE .

Необхідно враховувати, що у версіях до 3.23.17 оператор INNER JOIN не приймає параметр join\_condition !

Наявність останньої з наведених вище конструкцій вираження LEFT OUTER JOIN обумовлено лише вимогами сумісності з ODBC:

Замість посилання на таблицю може використовуватися псевдонім, який присвоюється за допомогою виразів tbl\_name AS alias\_name або tbl\_name alias\_name :

mysql> SELECT t1.name, t2.salary FROM employee AS t1, info AS t2

WHERE t1.name = t2.name;

Умовний оператор ON являє собою умова в будь-якій формі з числа тих, які можна використовувати у виразі WHERE .

Якщо запис для правої таблиці в частинах ON або USING в LEFT JOIN не знайдено, то для даної таблиці використовується рядок, в якій всі стовпці встановлені в NULL . Цю можливість можна застосовувати для знаходження результатів в таблиці, яка не має еквівалента в іншій таблиці:

mysql> SELECT table1. \* FROM table1

LEFT JOIN table2 ON table1.id = table2.id

WHERE table2.id IS NULL;

Цей приклад знаходить всі рядки в таблиці table1 з величиною id , яка не присутня в таблиці table2 (тобто всі рядки в table1 , для яких немає відповідних рядків у table2 ). Звичайно, це передбачає, що table2.id оголошений як NOT NULL . See section 5.2.6 Як MySQL оптимізує LEFT JOIN і RIGHT JOIN .

USING (column\_list) служить для зазначення списку стовпців, які повинні існувати в обох таблицях. Такий вираз USING , як:

A LEFT JOIN B USING (C1, C2, C3, ...)

семантично ідентично вираженню ON , наприклад:

A.C1 = B.C1 AND A.C2 = B.C2 AND A.C3 = B.C3, ...

Вираз NATURAL [LEFT] JOIN для двох таблиць визначається так, щоб воно було семантичним еквівалентом INNER JOIN або LEFT JOIN з виразом USING , в якому зазначені всі стовпці, наявні в обох таблицях.

INNER JOIN і ,(кома) є семантичними еквівалентами. Обидва здійснюють повне об'єднання використовуваних таблиць. Спосіб зв'язування таблиць зазвичай задається в умові WHERE .

RIGHT JOIN працює аналогічно LEFT JOIN . Для збереження переносимості коду між різними базами даних рекомендується замість RIGHT JOIN використовувати LEFT JOIN .

STRAIGHT\_JOIN ідентично JOIN , за винятком того, що ліва таблиця завжди читається раніше правої. Це вираз може використовуватися для тих (небагатьох) випадків, коли оптимізатор об'єднання своєму розпорядженні таблиці в неправильному порядку.

Починаючи з версії MySQL 3.23.12, можна давати MySQL вказівки про те, який індекс повинен використовуватися при витяганні інформації з таблиці. Ця можливість корисна, якщо оператор EXPLAIN (виводить інформацію про структуру та порядок виконання запиту SELECT ), показує, що MySQL використовує хибний індекс. Задаючи значення індексу в USE INDEX (key\_list), можна змусити MySQL застосовувати для пошуку запису тільки один із зазначених індексів. Альтернативне вираз IGNORE INDEX (key\_list) забороняє використання в MySQL даного конкретного індексу. Вирази USE / IGNORE KEY є синонімами для USE / IGNORE INDEX .

Оператор INSERT має наступну структуру:

INSERT [LOW\_PRIORITY | DELAYED] [IGNORE]

[INTO] tbl\_name [(col\_name, ...)]

VALUES (expression, ...), (...), ...

або

INSERT [LOW\_PRIORITY | DELAYED] [IGNORE]

[INTO] tbl\_name [(col\_name, ...)]

SELECT ...

або

INSERT [LOW\_PRIORITY | DELAYED] [IGNORE]

[INTO] tbl\_name

SET col\_name = expression, col\_name = expression, ...

Оператор INSERT вставляє нових рядків в існуючу таблицю. Форма даної команди INSERT ... VALUES вставляє рядки відповідно до точно вказаними в команді значеннями. Форма INSERT ... SELECT вставляє рядки, обрані з іншої таблиці або таблиць. Форма INSERT ... VALUES зі списком з декількох значень підтримується у версії MySQL 3.22.5 і пізніших. Синтаксис вираження col\_name = expression підтримується у версії MySQL 3.22.10 і пізніших.

tbl\_name задає таблицю, в яку повинні бути внесені рядки. Стовпці, для яких задані величини в команді, вказуються в списку імен стовпців або в частині SET :

Якщо не вказано список стовпців для INSERT ... VALUES або INSERT ... SELECT , то величини для всіх стовпців повинні бути визначені в списку VALUES () або в результаті роботи SELECT . Якщо порядок стовпців у таблиці невідомий, для його отримання можна використовувати DESCRIBE tbl\_name .

Будь стовпець, для якого явно не вказано значення, буде встановлений у своє значення за замовчуванням. Наприклад, якщо в заданому списку стовпців не вказані всі стовпці в даній таблиці, то чи не згадані стовпці встановлюються у свої значення за замовчуванням. Установка значень за замовчуванням описується в розділі section 6.5.3 Синтаксис оператора CREATE TABLE . В MySQL завжди передбачено значення за замовчуванням для кожного поля. Ця вимога `` нав'язано'' MySQL, щоб забезпечити можливість роботи як з таблицями, що підтримують транзакції, так і з таблицями, що не підтримують їх. Наша точка зору (розробників) полягає в тому, що перевірка вмісту полів повинна проводитися додатком, а не сервером баз даних.

Вираз expression може відноситься до будь-якому стовпцю, який раніше був внесений до списку значень. Наприклад, можна вказати наступне:

mysql> INSERT INTO tbl\_name (col1, col2) VALUES (15, col1 \* 2);

Але не можна вказати:

mysql> INSERT INTO tbl\_name (col1, col2) VALUES (col2 \* 2,15);

Якщо вказується ключове слово LOW\_PRIORITY , то виконання даної команди INSERT буде затримано доти, поки інші клієнти не завершать читання цієї таблиці. У цьому випадку дана клієнт повинен очікувати, поки дана команда вставки не будуть завершені, що в разі інтенсивного використання таблиці може зажадати значного часу. На противагу цьому команда INSERT DELAYED дозволяє даному клієнтові продовжувати операцію відразу. Синтаксис оператора INSERT DELAYED . Слід зазначити, що покажчик LOW\_PRIORITY зазвичай не використовується з таблицями MyISAM , оскільки при його вказівці стають неможливими паралельні вставки.

Якщо в команді INSERT з рядками, що мають багато значень, вказується ключове слово IGNORE , то всі рядки, що мають дублюються ключі PRIMARY або UNIQUE у цій таблиці, будуть проігноровані і не будуть внесені. Якщо не вказувати IGNORE , то дана операція вставки припиняється при виявленні рядка, що має дублюється значення існуючого ключа. Кількість рядків, внесених в дану таблицю, можна визначити за допомогою функції C API mysql\_info () .

Якщо MySQL був налаштований з використанням опції DONT\_USE\_DEFAULT\_FIELDS , то команда INSERT буде генерувати помилку, якщо явно не вказати величини для всіх стовпців, які вимагають у допустимих- NULL.

За допомогою функції mysql\_insert\_id можна знайти величину, застосовану для стовпця AUTO\_INCREMENT.

Якщо задається команда INSERT ... SELECT або INSERT ... VALUES зі списками з декількох значень, то для отримання інформації про даний запиті можна використовувати функцію C API mysql\_info () . Формат цієї інформаційної рядка наведено нижче:

Records: 100 Duplicates: 0 Warnings: 0

Duplicates показує кількість рядків, які не могли бути внесені, оскільки вони дублювали б значення деяких існуючих унікальних індексів. Покажчик Warnings показує число спроб внести величину в стовпець, який з якої-небудь причини опинився проблематичним. Попередження виникають при виконанні будь-якого з таких умов:

Внесення NULL в стовпець, який був оголошений, як NOT NULL . Даний стовпець встановлюється в значення, задане за замовчуванням.

Установка числового стовпця в значення, що лежить за межами його допустимого діапазону. Дана величина буде скорочуватися до відповідної кінцевої точки цього діапазону.

Занесення до числової стовпець такої величини, як '10 .34 a ' . Кінцеві дані видаляються і вноситься тільки числова частина. Якщо величина зовсім не має сенсу як число, то стовпець встановлюється у 0 .

Внесення у стовпці типу CHAR , VARCHAR , TEXT або BLOB рядка, яка перевершує максимальну довжину стовпчика. Дана величина буде скорочуватися до максимальної довжини стовпчика.

Внесення в стовпець дати або часу рядки, неприпустимою для даного типу стовпця. Цей стовпець встановлюється у нульову величину, відповідну даному типу.

Оператор UPDATE має наступну структуру:

UPDATE [LOW\_PRIORITY] [IGNORE] tbl\_name

SET col\_name1 = expr1 [, col\_name2 = expr2, ...]

[WHERE where\_definition]

[LIMIT #]

Оператор UPDATE оновлює стовпці відповідно до їх новими значеннями в рядках існуючої таблиці. У виразі SET вказується, які саме стовпці слід модифікувати і які величини повинні бути в них встановлені. У виразі WHERE , якщо воно присутнє, задається, які рядки підлягають оновленню. В інших випадках оновлюються всі рядки. Якщо задано вираз ORDER BY , то рядки будуть оновлюватися у зазначеному в ньому порядку.

Якщо вказується ключове слово LOW\_PRIORITY , то виконання даної команди UPDATE затримується доти, поки інші клієнти не завершать читання цієї таблиці.

Якщо вказується ключове слово IGNORE , то команда поновлення не буде перервана, навіть якщо при оновленні виникне помилка дублювання ключів. Рядки, через які виникають конфліктні ситуації, оновлені не будуть.

Якщо доступ до колонки із зазначеного вираження здійснюється по аргументу tbl\_name , то команда UPDATE використовує для цього стовпця його поточне значення. Наприклад, наступна команда встановлює стовпець age в значення, на одиницю більше його поточної величини:

mysql> UPDATE persondata SET age = age +1;

Значення команда UPDATE присвоює зліва направо. Наприклад, наступна команда дублює стовпець age , потім інкрементує його:

mysql> UPDATE persondata SET age = age \* 2, age = age +1;

Якщо стовпець встановлюється в його поточне значення, то MySQL помічає це і не оновлює його.

Команда UPDATE повертає кількість фактично змінених рядків. У версії MySQL 3.22 і більш пізніх функція C API mysql\_info () повертає кількість рядків, які були знайдені і оновлені, і кількість попереджень, що мали місце при виконанні UPDATE .

У версії MySQL 3.23 можна використовувати LIMIT # , щоб переконатися, що було змінено тільки задану кількість рядків.

Оператор DELETE має наступну структуру:

DELETE [LOW\_PRIORITY | QUICK] FROM table\_name

[WHERE where\_definition]

[ORDER BY ...]

[LIMIT rows]

або

DELETE [LOW\_PRIORITY | QUICK] table\_name [. \*] [, Table\_name [. \*] ...]

FROM table-references

[WHERE where\_definition]

або

DELETE [LOW\_PRIORITY | QUICK]

FROM table\_name [. \*], [Table\_name [. \*] ...]

USING table-references

[WHERE where\_definition]

Оператор DELETE видаляє з таблиці table\_name рядки, що задовольняють заданим в where\_definition умовам, і повертає число віддалених записів.

Якщо оператор DELETE запускається без визначення WHERE , то видаляються всі рядки. При роботі в режимі AUTOCOMMIT це буде аналогічно використанню оператора TRUNCATE . В MySQL 3.23 оператор DELETE без визначення WHERE поверне нуль як число видалених записів.

Якщо дійсно необхідно знати число видалених записів при видаленні всіх рядків, і якщо припустимі втрати в швидкості, то можна використовувати команду DELETE в такій формі:

mysql> DELETE FROM table\_name WHERE 1> 0;

Слід враховувати, що ця форма працює набагато повільніше, ніж DELETE FROM table\_name без вираження WHERE , оскільки рядки видаляються по черзі по одній.

Якщо вказано ключове слово LOW\_PRIORITY , виконання даної команди DELETE буде затримано доти, поки інші клієнти не завершать читання цієї таблиці.

Якщо заданий параметр QUICK , то вказівник таблиці при виконанні видалення не об'єднуватиметься індекси - у деяких випадках це може прискорити дану операцію.

У таблицях MyISAM видалені записи зберігаються у зв'язаному списку, а наступні операції INSERT повторно використовують місця, де розташовувалися видалені записи. Щоб повернути не використаний простір і зменшити розмір файлів, можна застосувати команду OPTIMIZE TABLE або утиліту myisamchk для реорганізації таблиць. Команда OPTIMIZE TABLE простіше, але утиліта myisamchk працює швидше.

Перший з числа наведених на початку даного розділу багатотабличних формат команди DELETE підтримується, починаючи з MySQL 4.0.0. Другий багатотабличних формат підтримується, починаючи з MySQL 4.0.2.

Ідея полягає в тому, що видаляються тільки збережені рядки з таблиць, перераховані перед виразами FROM або USING . Це дозволяє видаляти одночасно рядки з декількох таблиць, а також використовувати для пошуку додаткові таблиці.

Символи . \* після імен таблиць потрібні тільки для сумісності з Access:

DELETE t1, t2 FROM t1, t2, t3 WHERE t1.id = t2.id AND t2.id = t3.id

або

DELETE FROM t1, t2 USING t1, t2, t3 WHERE t1.id = t2.id AND t2.id = t3.id

У попередньому випадку просто видалені збігаються рядки з таблиць t1 і t2

Вираз ORDER BY і використання декількох таблиць в команді DELETE підтримується в MySQL 4.0.

Якщо застосовується вираз ORDER BY , то рядки будуть видалені в зазначеному порядку. Насправді цей вислів корисно тільки в поєднанні з LIMIT. Наприклад:

DELETE FROM somelog

WHERE user = 'jcole'

ORDER BY timestamp

LIMIT 1

Даний оператор видалить найстаріший запис (за timestamp ), в рядку який вказаній у вираженні WHERE .

Специфічна для MySQL опція LIMIT для команди DELETE вказує серверу максимальну кількість рядків, які слід видалити до повернення управління клієнту. Ця опція може використовуватися для гарантії того, що дана команда DELETE не зажадає занадто багато часу для виконання. Можна просто повторювати команду DELETE до тих пір, поки кількість вилучених рядків буде менша, ніж величина LIMIT.

* 1. **Створення структури бази даних**

Для того, щоб спроектувати структуру бази даних необхідно спочатку чітко визначити які функції повинна виконувати наша програма.

* Авторизація при вході в програму
* Додавання записів
* Видалення записів
* Редагування записів
* Надання різних прав доступу в програмі
* Нарахування зарплати
* Прихід коштів в різних валютах

На основі цих критеріїв спроектуємо базу даних, яка буде містити інформацію про клієнтів, проекти, працівників, прихід коштів, витрати, зарплату.

Розроблена структура бази даних показана на рисунку 2.2.

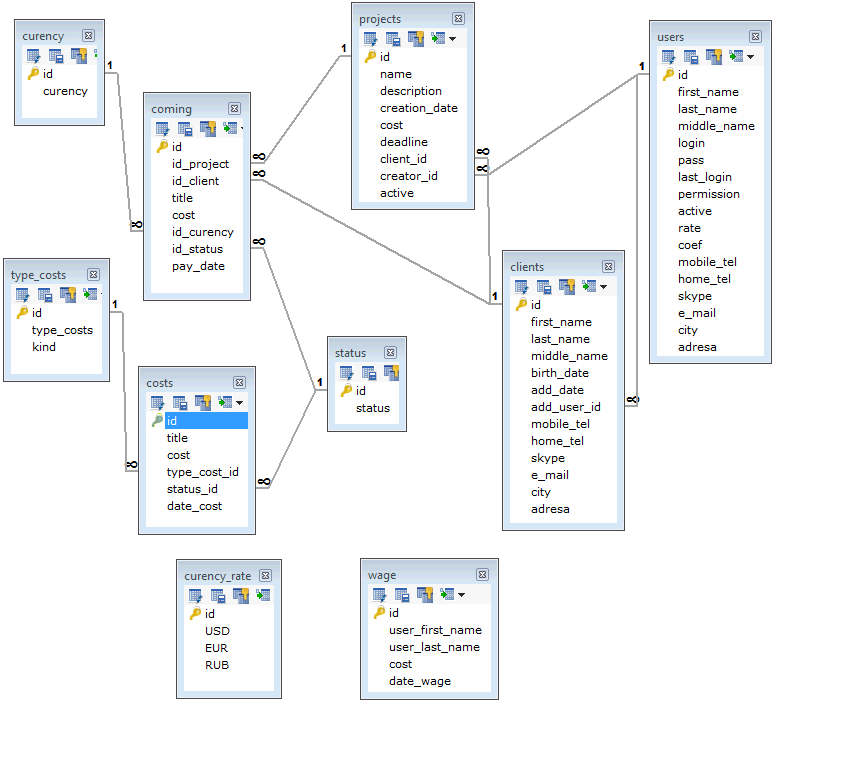


Рисунок 2.2 – Розроблена структура бази даних проекту

Таблиця clients містить в собі інформацію про клієнтів, їх ім’я, прізвище, по батькові та контактні дані. В полі add\_data зберігається інформація про те, коли дані про клієнта були добавлені в базу даних. В полі add\_user\_id зберігається інформація про користувача, який додав ці дані.

Таблиця 2.1 – Клієнти

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип поля** | **Призначення** |
| id | int(11) | ключове поле |
| first\_name | varchar(100) | прізвище |
| last\_name | varchar(100) | ім’я |
| middle\_name | varchar(100) | по батькові |
| birthday | date | дата народження |
| add\_date | date | дата додання |
| add\_user\_id | int(11) | хто додав |
| mobile\_tel | varchar(150) | мобільний телефон |
| home\_tel | varchar(100) | домашній телефон |
| skype | varchar(100) | skype |
| e\_mail | varchar(250) | e-mail |
| city | varchar(200) | місто |
| adresa | varchar(250) | адреса |

Таблиця projects містить інформацію про проекти замовлені клієнтами компанії Тімтек. В ній зберігаються такі дані як: скорочена назва проекту, опис проекту або технічне завдання, дата початку проекту та дата закінчення проекту, також ціна проекту та прізвище клієнта який його замовив. В полі active позначається чи проект активний чи неактивний.

Таблиця 2.2 – Проекти

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип поля** | **Призначення** |
| id | int(11) | ключове поле |
| name | varchar(250) | назва проекту |
| description | text | опис |
| creation\_date | date | дата початку |
| cost | float | вартість |
| deadline | date | дата закінчення |
| client\_id | int(11) | зв’язок з клієнтами |
| creator\_id | int(11)) | зв’язок з користувачами |
| active | tinyint(4) | активний/неактивний |

Таблиця users містить дані про працівників (користувачів програми) компанії,такі як: прізвище, ім’я, по батькові, контактні дані, логін і пароль для авторизації в програмі. Також в цій таблиці знаходяться дані для розрахунку зарплати працівникам. В полі permission міститься інформація про привілегії користувача (права доступу в програмі). Поле last\_login зберігає дату, коли користувач останній раз заходив в програму, а поле active вказує чи даний користувач активований адміністратором. Якщо він неактивний, то не зможе авторизуватись в програмі. Поля rate та coef зберігають дані про мінімальну ставку зарплати в місяць та коефіцієнт від прибутку який сумується до зарплати.

Таблиця 2.3 – Користувачі

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип поля** | **Призначення** |
| id | int(11) | ключове поле |
| first\_name | varchar(100) | прізвище |
| last\_name | varchar(100) | ім’я |
| middle\_name | varchar(100) | по батькові |
| login | varchar(45) | логін |
| pass | varchar(45) | пароль |
| last\_login | date | дата останньої авторизації |
| permission | int(11) | права доступу |
| active | tinyint(4) | активний/неактивний |
| rate | int(10) | мінімальна ставка |
| coef | float | коефіцієнт до зарплати |
| mobile\_tel | varchar(150) | мобільний телефон |
| home\_tel | varchar(100) | домашній телефон |
| skype | varchar(100) | skype |
| e\_mail | varchar(250) | e-mail |
| city | varchar(200) | місто |
| adresa | varchar(250) | адреса |

Таблиця coming зберігає інформацію про прихід коштів компанії. Поле id\_project містить назву проекта на який було оплачено, поле id\_client це прізвище клієнта який здійснив оплату, cost поле в якому вказана сума приходу, id\_curency поле де вказується в якій валюті поступили кошти, id\_status – статус оплати проекту(оплачено, не оплачено, заборговано, електронний переказ), pay\_date поле із датою приходу коштів.

Таблиця 2.4 – Прихід

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип поля** | **Призначення** |
| id | int(11) | ключове поле |
| id\_project | int(11) | зв’язок з проектами |
| id\_client | int(11) | зв’язок з клієнтами |
| title | varchar(250) | примітка |
| cost | float | сума |
| id\_curency | int(11) | зв’язок з типами валют |
| id\_status | int(11)) | зв’язок з статусом |
| pay\_date | date | дата приходу |

В таблиці costs міститься інформація про витрати компанії. Поле title це примітка, тут записують на що було потрачено кошти. В полі cost вказується сума, type\_cost\_id це полез типами витрат (наприклад інтернет, канцтовари, комунальні послуги та ін.). Поле type\_cost\_id зв’язане з таблицею type\_costs в якій є поле kind. В цьому полі містяться дані про вид витрат (регулярні і не регулярні).

Таблиця 2.5 – Витрати

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип поля** | **Призначення** |
| id | int(11) | ключове поле |
| title | varchar(250) | примітка |
| cost | float | сума |
| type\_cost\_id | int(11) | зв’язок з типами витрат |
| status\_id | int(11)) | зв’язок з статусом |
| date\_cost | date | дата витрат |

Таблиця curency містить інформацію про валюти. Вона зв’язана з таблицею coming в якій по id ми вибираємо тип валют. А в таблиці currency\_rate знаходяться дані про курс валют.

Таблиця wage зберігає інформацію про нараховану зарплату працівникам. В полі user\_first\_name зберігається прізвище працівника, в user\_last\_name ім’я, в полі cost – записується сума заробітної плати, а в полі date\_wage - дата нарахування.

Таблиця 2.6 – Зарплата

|  |  |  |
| --- | --- | --- |
| **Назва поля** | **Тип поля** | **Призначення** |
| id | int(11) | ключове поле |
| user\_first\_name | varchar(100) | прізвище працівника |
| user\_last\_name | varchar(100) | ім’я працівника |
| cost | float | сума |
| date\_wage | date | дата нарахування |

* 1. **Структура проекту**

Файл проекту являє собою програму, написану мовою Object Pascal і призначену для обробки компілятором. Ця програма автоматично створюється Delphi і містить лише кілька рядків:

program Project1;

uses

Forms,

Unit1 in 'Unit1.pas' {Form1},

...

Unit9 in 'Unit9.pas' {Form9};

{$R \*.res}

begin

Application.Initialize;

Application.MainFormOnTaskbar := True;

Application.Title := 'TimTek';

Application.CreateForm(TForm1, Form1);

…

Application.CreateForm(TForm9, Form9);

Application.Run;

end.

Як ми бачимо в нашому проекті є 9 форм(вікон), які створюються після ініціалізації. Кожна форма це окремий модуль програми. Всі вони описані в розділі uses наступним кодом Unit1 in 'Unit1.pas' {Form1}. Програмний код модулів наведений в додатку А.

Оскільки програма працює з базою даних, то практично всі дії, які виконуються в ній це запити до бази даних. Приклад процедури, яка робить запит для вибірки клієнтів:

procedure select\_clients (Query :TSQLQuery);

begin

Query.SQL.Clear;

Query.SQL.Add('SELECT clients.id, clients.first\_name,clients.last\_name, ');

Query.SQL.Add('clients.middle\_name,clients.birth\_date, ');

Query.SQL.Add('clients.mobile\_tel,clients.home\_tel,clients.skype, ');

Query.SQL.Add('clients.e\_mail,clients.city,clients.adresa, ');

Query.SQL.Add(' clients.add\_date, users.login as add\_user\_name FROM ');

Query.SQL.Add('clients INNER JOIN users ON (clients.add\_user\_id = users.id)');

Query.ExecSQL();

end;

Вище наведений код після виконання процедури передає запит на MySQL сервер. Сервер виконує запит і повертає вибрані значення в компонент для відображення таблиць DBGrid.

Таким чином більшість процедур в модулях програми та дій в програмі, а саме: авторизація для входу в програму, додавання, редагування, видалення записів, обчислення грошових доходів і витрат, нарахування зарплати працівникам, фільтри пошуку виконуються запитом до бази даних.

1. **РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ ВЕДЕННЯ ОБЛІКУ КЛІЄНТІВ ТА ГРОШОВИХ ТРАНЗАКЦІЙ ПІДПРИЄМСТВА**
   1. **Створення бази даних та забезпечення зв’язків між її елементами**

Для створення бази даних вибрано MySQL та систему управління SQLyog Ultimate. Всі таблиці в базі даних типу InnoDB.

InnoDB — це потужний механізм (рушій) для зберігання даних. Застосування InnoDB дозволяє використання базою даних таких функцій, як транзакції, зовнішні ключі. Він також сумісний з ACID.

У цьому механізмі є два способи для зберігання даних: файл або група файлів, загальних для всіх баз даних і таблиць, або один файл даних для кожної таблиці. Інші важливі особливості InnoDB: блокування на рівні рядків, можливість стиснення даних.

Створення таблиць в SQLyog можливо 2 способами: у візуальному режимі і за допомогою SQL-запиту.

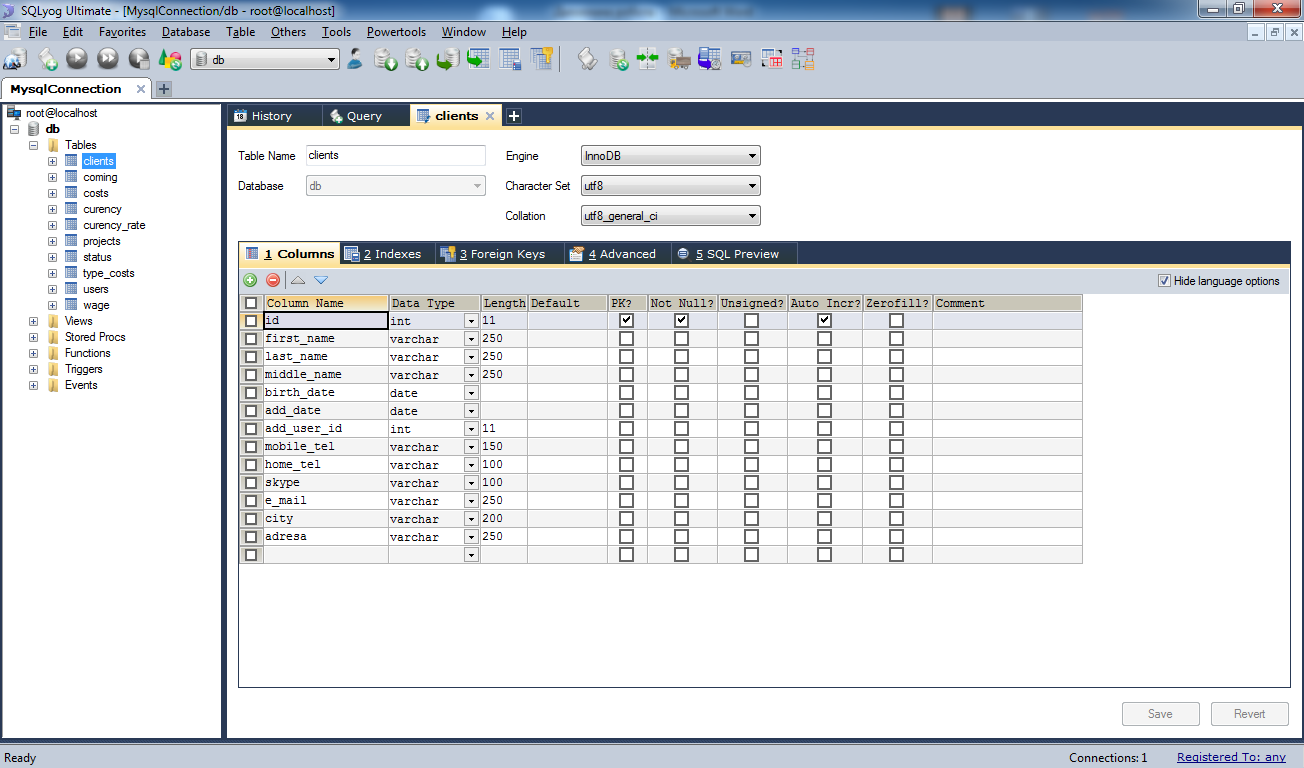


Рисунок 3.1 – Створення таблиць у візуальному режимі

Наступний код показує можливість створення таблиці coming через SQL-запит:

CREATE TABLE `coming` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`id\_project` int(11) DEFAULT NULL,

`id\_client` int(11) DEFAULT NULL,

`title` varchar(250) DEFAULT NULL,

`cost` float DEFAULT NULL,

`id\_curency` int(11) DEFAULT NULL,

`id\_status` int(11) DEFAULT NULL,

`pay\_date` date DEFAULT NULL,

PRIMARY KEY (`id`),

KEY `id\_project` (`id\_project`),

KEY `id\_client` (`id\_client`),

KEY `id\_curency` (`id\_curency`),

CONSTRAINT `coming\_ibfk\_3` FOREIGN KEY (`id\_project`) REFERENCES `projects` (`id`),

CONSTRAINT `coming\_ibfk\_4` FOREIGN KEY (`id\_client`) REFERENCES `clients` (`id`),

CONSTRAINT `coming\_ibfk\_5` FOREIGN KEY (`id\_curency`) REFERENCES `curency` (`id`));

Запустивши запит на виконання, в базі даних створиться нова таблиця з іменем coming (прихід), а також створяться зв'язки між таблицею coming та таблицями projects (проекти), clients(клієнти), curency(тип валюти).

Зв'язки потрібні для того, щоб використовувати дані однієї таблиці в іншій. В даному випадку зв’язавши ці таблиці ми можемо при додаванні запису в таблицю coming (прихід) вибирати назву проекту на який прийшли кошти з таблиці projects (проекти), прізвище клієнта, який оплатив проект з таблиці clients (клієнти), а також тип валюти з таблиці curency.

* 1. **Розробка функціональних можливостей програми**
     1. *Авторизація для входу в програму*

Так як програма працює з грошовими транзакціями підприємства, то для захисту інформації розроблено форму авторизації для входу в програму.

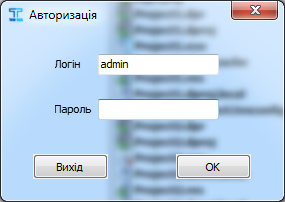


Рисунок 3.2 – вигляд вікна авторизації

Всі працівники, які мають доступ до програми поділені на дві категорії, що надають обмеження прав в програмі: користувачів і адміністраторів. Адміністраторам доступні всі можливості програми, а користувачам тільки перегляд деякої інформації.

Якщо це перший запуск програми, то вона автоматично створить в БД адміністратора із логіном – admin і паролем – admin. Після входу адміністратор має можливість його змінити.

Перший запуск і перевірку логіна і пароля програма здійснює по натисненню кнопки ОК, наступним кодом:

ClientDataSet1.Active:=false;

SQlQuery1.Active := false;

SQLQuery1.SQL.Clear;

SQLQuery1.SQL.Add('select \* from db.users');

SQLQuery1.Active:= true;

ClientDataSet1.Active:=true;

if DataSource1.DataSet.Eof then

begin

SQLQuery1.SQL.Clear;

SQLQuery1.SQL.Add('insert into db.users (login, pass, active, rate, coef, ');

SQLQuery1.SQL.Add('permission) values("admin", "admin", 1, 1000, 0, 1)');

SQLQuery1.ExecSQL();

Form2.ClientDataSet3.Refresh;

Form2.Show;

end

else

begin

ClientDataSet1.Active:=false;

SQlQuery1.Active := false;

SQLQuery1.SQL.Clear;

SQLQuery1.SQL.Add('select id, login, pass, last\_login, active, permission ');

SQLQuery1.SQL.Add('from users where ');

SQLQuery1.SQL.Add('(login ='''+Trim(Form1.edit1.text)+''' and ');

SQLQuery1.SQL.Add('pass ='''+Trim(Form1.MaskEdit1.Text)+''' and active = 1)');

SQLQuery1.Active:= true;

ClientDataSet1.Active:=true;

if not SQLQuery1.IsEmpty then

begin

id\_user := strtoint(DataSource1.DataSet.FieldByName('id').Value);

if DataSource1.DataSet.FieldByName('permission').AsInteger = 1 then

begin

Form2.Show;

Form1.Hide;

end;

Except

MessageDlg('Неправильно заповнені поля',mtError,[mbOK],0);

end;

Якщо введеного логіна і пароля немає в базі даних то програма видає вікно з помилкою.

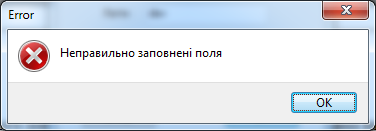


Рисунок 3.3 – вікно з помилкою авторизації

* + 1. *Головне меню програми*

Головне меню програми складається з трьох пунктів, які в свою чергу складаються з підпунктів:

1. Файл
2. Звіти

* Вода
* Електрика

1. Завершити сеанс
2. Вихід
3. Налаштування
4. Курс валют
5. Справка
6. Help

При натисканні на підпункт Завершити сеанс, програма поверне нас до вікна авторизації. Виходити з програми необов’язково через меню, для цього в програмі зарезервовано комбінацію клавіш Alt+X.

Підпункт Курс валют відкриє вікно для зміни курсу валют, який дуже важливий для обчислень.

Help відкриє на файл справки, в якому описано як користуватися програмою. Також файл справки можна запустити натиснувши клавішу F1.

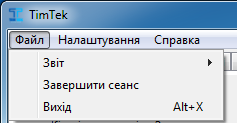


Рисунок 3.4 – головне меню програми

* + 1. *Робоче вікно програми та його основні можливості*

Інтерфейс програми побудований так, що вся інформація і весь функціонал програми знаходиться в одному вікні, але поділений на кілька вкладок. Фактично кожна інша вкладка це одна таблиця в нашій базі даних.

Після успішної авторизації відкриється головне робоче вікно програми.

Перша вкладка служить тільки для відображення інформації, а саме: кількість клієнтів підприємства, кількість проектів (активних і неактивних), відображає загальний прихід коштів та прихід коштів за місяць, обчислює регулярні витрати та нерегулярні витрати за поточний місяць.



Рисунок 3.5 – інформаційна вкладка

Наступна вкладка Клієнти. Тут відображається вся інформація про клієнтів компанії ТімТек.

Інтерфейс вкладки побудований наступним чином: справа розміщена панель з трьома кнопками (додати, редагувати, видалити запис з БД), вверху розміщена панель з фільтрами пошуку по таблиці clients, а в середині відображається сама таблиця з даними.

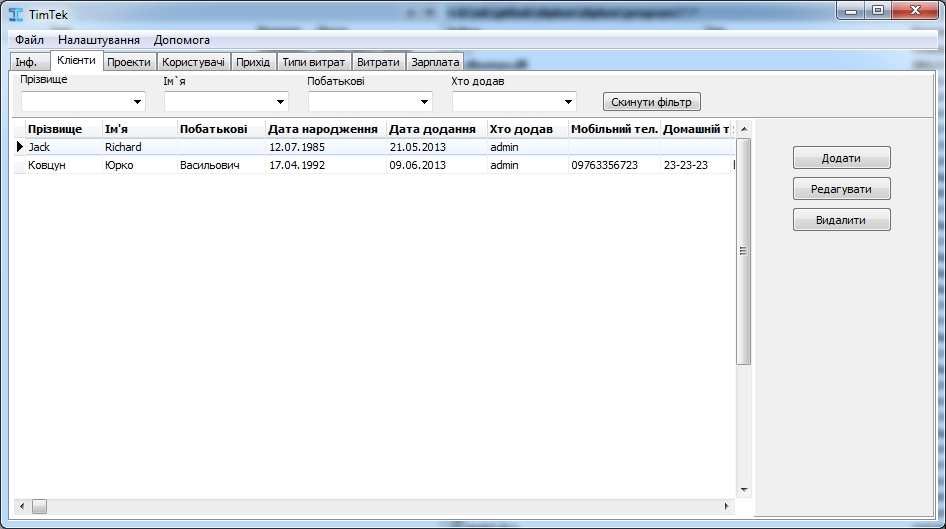


Рисунок 3.6 – вигляд вкладки Клієнти

Для того, щоб видалити запис з бази даних потрібно виділити запис в таблиці і нажати на кнопку Видалити, після чого з’явиться вікно для підтвердження видалення.

Кнопки Додати та Редагувати відкривають одну форму, яка служить для обох операцій, тільки змінюються функціональні кнопки. Додаються и редагуються записи за допомогою SQL-запитів.

Приклад процедури додання нового клієнта:

procedure TForm3.Button1Click(Sender: TObject);

begin

try

Form2.SQLQuery1.SQL.Clear;

Form2.SQLQuery1.SQL.Add('INSERT INTO db.clients (first\_name, last\_name, middle\_name');

Form2.SQLQuery1.SQL.Add(', birth\_date, add\_date, add\_user\_id, mobile\_tel, ');

Form2.SQLQuery1.SQL.Add('home\_tel, skype, e\_mail, city, adresa) ');

Form2.SQLQuery1.SQL.Add(' values(:f\_n, :l\_n, :m\_n, :b\_d, :a\_d, :a\_u\_i, ');

Form2.SQLQuery1.SQL.Add(':m\_t, :h\_t, :s, :e, :c, :a) ');

Form2.SQLQuery1.Params.ParamByName('f\_n').Value := Edit1.Text;

Form2.SQLQuery1.Params.ParamByName('l\_n').Value := Edit2.Text;

Form2.SQLQuery1.Params.ParamByName('m\_n').Value := Edit3.Text;

Form2.SQLQuery1.Params.ParamByName('b\_d').Value:=

FormatDateTime('yyyy-mm-dd', DateTimePicker1.Date);

Form2.SQLQuery1.Params.ParamByName('a\_d').Value:=

FormatDateTime('yyyy-mm-dd',DateTimePicker2.Date);

Form2.SQLQuery1.Params.ParamByName('a\_u\_i').Value := form1.id\_user;

Form2.SQLQuery1.Params.ParamByName('m\_t').Value := Edit4.Text;

Form2.SQLQuery1.Params.ParamByName('h\_t').Value := Edit5.Text;

Form2.SQLQuery1.Params.ParamByName('s').Value := Edit6.Text;

Form2.SQLQuery1.Params.ParamByName('e').Value := Edit7.Text;

Form2.SQLQuery1.Params.ParamByName('c').Value := Edit8.Text;

Form2.SQLQuery1.Params.ParamByName('a').Value := Edit9.Text;

Form2.SQLQuery1.ExecSQL();

select\_clients(Form2, Form2.SQLQuery1);

Form2.ClientDataSet1.Refresh;

Except

MessageDlg('Неправильно заповнені поля',mtError,[mbOK],0);

end;

Form3.Hide;

end;

Як ми бачимо виконується запит INSERT до БД, який через параметри читає значення текстових полів.

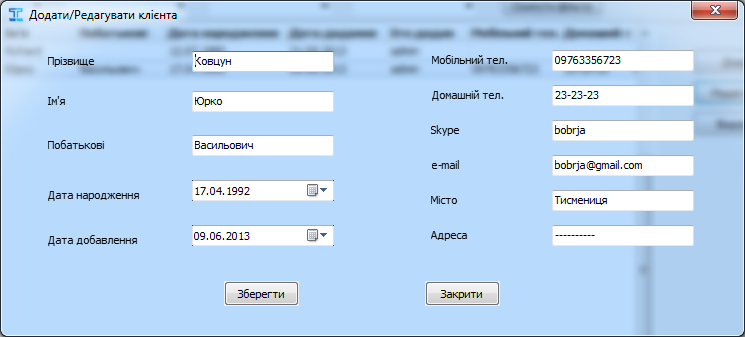


Рисунок 3.7 – вікно додання та редагування клієнтів

Всі наступні вкладки: проекти, користувачі, прихід, типи витрат, витрати мають такий самий інтерфейс та виконують такі ж функції, як і на вкладці клієнти. Єдина відмінність є тільки при додаванні нового користувача. Оскільки тут додається логін та пароль для входу в програму, то при доданні програма перевіряє чи немає такого логіна в базі даних.

Вкладка Зарплата має деякі відмінності від інших. Справа на панелі є тільки дві кнопки: зарплата та видалити. Кнопка Видалити видаляє запис з бази даних, а кнопка Зарплата рахує зарплату працівникам за місяць і додає нові записи в таблицю. Працівників яким потрібно нарахувати зарплату програма вибирає з таблиці users по таким критеріям:

* Працівник є Активний
* Щомісячна ставка більша нуля
* Коефіцієнт до зарплати відмінний від нуля

Після вибірки програма бере прибуток коштів за місяць, віднімає регулярні витрати, витрати за місяць, суму ставок. Далі цей залишок множимо на коефіцієнт кожного працівника і додаєм до основної ставки.

* + 1. *Налаштування програми для підключення до БД*

Для того, щоб програма знала адрес, логін та пароль підключення до MySQL-сервера потрібно запустити файл налаштувань. Після запуску якого нам відкриється вікно для вводу даних підключення.

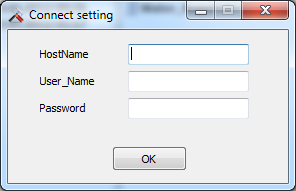


Рисунок 3.8 – вигляд вікна налаштувань

Після натиснення на кнопку ОК програма збереже дані підключення в файл з розширенням ini.

Введена інформація перед збереженням в файл шифрується, і тільки після того зберігається.

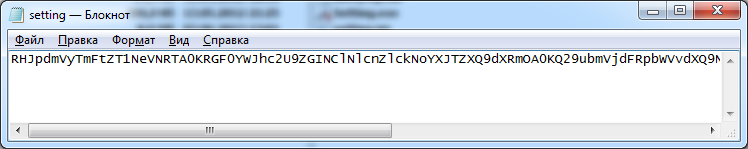


Рисунок 3.9 – зашифрований файл налаштувань

Основна програма під час запуску зчитує дані налаштування із зашифрованого файла, після чого проходить підключення до бази даних.

1. **ОХОРОНА ПРАЦІ**
   1. **Законодавче та нормативно-правове забезпечення охорони праці**

Компанія, в якій я проходив переддипломну практику працює в сфері ІТ-технологій , займається розробкою програмного забезпечення, web-розробкою, технічною підтримкою та ІТ-безпекою.

Оскільки компанія «ТімТек» працює в сфері ІТ, то працівники цієї компанії практично весь час працюють з персональним комп’ютером. Тому для даного підприємства є обов`язковим виконання наступних нормативно-правових актів з охорони праці:

- НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно-обчислювальних машин

- НПАОП 0.00-4.12-05 Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці;

- НПАОП 0.00-4.15-98 Положення про розробку інструкцій з охорони праці;

- НПАОП 0.00-4.21-04 Типове положення про службу охорони праці;

- НПАОП 0.00-4.29-97 Типове положення про кабінет охорони праці;

- НПАОП 0.00-6.07-04 Порядок ведення обліку даних про технічний стан машин, механізмів, устаткування підвищеної небезпеки;

- НПАОП 0.00-6.23-92 Про порядок атестації робочих місць за умовами праці;

- НПАОП 0.00-7.06-94 Єдина державна система показників обліку умов та безпеки праці;

Правила охорони праці під час експлуатації електронно-обчислювальних машин поширюються на всіх суб'єктів господарювання незалежно від форм власності, які у своїй діяльності здійснюють роботу, пов'язану з електронно-обчислювальними машинами з відеодисплейними терміналами, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і периферійними пристроями.

Вимоги Правил є обов'язковими для роботодавців, операторів електронно-обчислювальних машин, операторів комп'ютерного набору, операторів комп'ютерної верстки та працівників інших професій, які у своїй роботі застосовують ЕОМ з ВДТ і ПП.

Найбільш повним нормативним документом щодо забезпечення охорони праці користувачів ПК є "Державні санітарні норми і правила роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин".

Дотримання вимог цих правил може значно знизити наслідки несприятливої дії на працівників шкідливих та небезпечних факторів, які супроводжують роботу з відеодисплейними матеріалами, зокрема можливість зорових, нервово-емоційних переживань, серцево-судинних захворювань. Виходячи з цього, підприємство має забезпечити гігієнічні й ергономічні вимоги щодо організації робочих приміщень для експлуатації ВДТ, робочого середовища, робочих місць з ВДТ, режиму праці і відпочинку при роботі з ВДТ тощо, які викладені у Правилах.

Природне освітлення в приміщеннях з ВДТ має здійснюватися через вікна, орієнтовані переважно на північ або північний схід і забезпечувати коефіцієнт природної освітленості не нижче ніж 1,5 %. Для захисту від прямих сонячних променів, які створюють прямі та відбиті відблиски з поверхні екранів ПК і клавіатури повинні бути передбачені сонцезахисні пристрої, вікна повинні мати жалюзі або штори.

Велике значення також надається правильній робочій позі працівника. При незручній робочій позі можуть з'явитися болі в м'язах, суглобах і сухожиллях. Вимоги до робочої пози наступні:

* голова не повинна бути нахилена більш ніж на 20°,
* плечі повинні бути розслаблені,
* лікті - під кутом 80°...100°,
* передпліччя й кисті рук - у горизонтальному положенні.

Причина неправильної пози користувачів обумовлена наступними факторами: немає гарної підставки для документів, клавіатура перебуває занадто високо, а документи - низько, нікуди покласти руки й кисті, недостатній простір для ніг.

З метою подолання зазначених недоліків даються загальні рекомендації: краще пересувна клавіатура; повинні бути передбачені спеціальні пристосування для регулювання висоти стола, клавіатури й екрана, а також підставка для рук.

Істотне значення для продуктивної і якісної роботи на комп'ютері мають розміри знаків, щільність їхнього розміщення, контраст і співвідношення яркостей символів і екрану. Якщо відстань від очей оператора до екрана дисплея становить 60...80 см, то висота знака повинна бути не менш 3мм, оптимальне співвідношення ширини й висоти знака становить 3:4, а відстань між знаками - 15...20% їхньої висоти. Співвідношення яскравості тла екрана й символів - від 1:2 до 1:15.

Під час користування комп'ютером фахівці радять установлювати монітор на відстані 50-60 см від очей. Також вважають, що верхня частина монітору повинна бути на рівні очей або трохи нижче. Коли людина дивиться прямо перед собою, його очі відкриваються ширше, ніж коли він дивиться униз. За рахунок цього площа огляду значно збільшується, викликаючи зневоднювання очей. До того ж якщо екран установлений високо, а очі широко відкриті, порушується функція моргання. Це значить, що очі не закриваються повністю, не обмиваються слізною рідиною, не одержують достатнього зволоження, що приводить до їхньої швидкої стомлюваності.

Створення сприятливих умов праці та правильне естетичне оформлення робочих місць на виробництві мають велике значення як для полегшення праці, так і для підвищення її привабливості, що позитивно впливає на продуктивність.

* 1. **Організація роботи з охорони праці на підприємстві**

Підприємство зобов'язане створити на робочому місці умови праці відповідно до нормативно-правових актів, а також забезпечити додержання вимог законодавства щодо прав працівників у галузі охорони праці.

Із цією метою компанія має забезпечити функціонування системи управління охороною праці, а саме:

* створює відповідні служби і призначає посадових осіб, які забезпечують вирішення конкретних питань охорони праці, затверджує інструкції про їхні обов'язки, права та відповідальність за виконання покладених на них функцій, а також контролює їх додержання;
* розробляє за участю сторін колективного договору і реалізує комплексні заходи для досягнення встановлених нормативів та підвищення існуючого рівня охорони праці;
* забезпечує виконання необхідних профілактичних заходів відповідно до обставин, що змінюються;
* впроваджує прогресивні технології, досягнення науки і техніки, засоби механізації та автоматизації виробництва, вимоги ергономіки, позитивний досвід з охорони праці тощо;
* забезпечує належне утримання будівель та споруд, виробничого обладнання та устаткування, моніторинг за їх технічним станом;
* забезпечує усунення причин, що призводять до нещасних випадків, професійних захворювань, та здійснення профілактичних заходів, визначених комісіями за підсумками розслідування цих причин;
* організовує проведення аудиту охорони праці, лабораторних досліджень умов праці, оцінку технічного стану виробничого обладнання та устаткування, атестацій робочих місць на відповідність нормативно-правовим актам з охорони праці в порядку і строки, що визначаються законодавством, та за їх підсумками вживає заходів з усунення небезпечних і шкідливих для здоров'я виробничих факторів;
* розробляє і затверджує положення, інструкції, інші акти з охорони праці, що діють у межах підприємства та встановлюють правила виконання робіт і поведінки працівників на території підприємства, у виробничих приміщеннях, на будівельних майданчиках, робочих місцях відповідно до нормативно-правових актів з охорони праці, забезпечує безоплатно працівників нормативно-правовими актами підприємства з охорони праці;
* здійснює контроль за додержанням працівником технологічних процесів, правил поведінки з машинами, механізмами, устаткуванням та іншими засобами виробництва, використанням засобів колективного та індивідуального захисту, виконанням робіт відповідно до вимог з охорони праці;
* організовує пропаганду безпечних методів праці та співробітництво з працівниками у галузі охорони праці.

Підприємство несе безпосередню відповідальність за порушення нормативно-правових актів з охорони праці.

Служба охорони праці створюється на підприємстві з кількістю працівників 50 і більше. Оскільки в компанії «ТімТек» працює менше 20 осіб, для виконання функцій служби охорони праці залучаються сторонні спеціалісти на договірних засадах.

Спеціалісти служби охорони праці у разі виявлення порушень охорони праці мають право:

* видавати керівникам структурних підрозділів підприємства обов'язкові для виконання приписи щодо усунення наявних недоліків, одержувати від них необхідні відомості, документацію і пояснення з питань охорони праці;
* вимагати відсторонення від роботи осіб, які не пройшли передбачених законодавством медичного огляду, навчання, інструктажу, перевірки знань і не мають допуску до відповідних робіт або не виконують вимог нормативно-правових актів з охорони праці;
* зупиняти роботу виробництва, дільниці, машин, механізмів, устаткування та інших засобів виробництва у разі порушень, які створюють загрозу життю або здоров'ю працівників;
* надсилати роботодавцю подання про притягнення до відповідальності працівників, які порушують вимоги щодо охорони праці.

Ліквідація служби охорони праці допускається тільки у разі ліквідації підприємства чи припинення використання найманої праці фізичною особою.

* 1. **Основні причини нещасних випадків на виробництві та профзахворювань і заходи щодо їх попередження**

За весь час існування компанії «ТімТек» не було жодного нещасного випадку. Також, ніхто із працівників компанії не хворів на професійні захворювання.

«Комп'ютерні» хвороби ще досить молоді і недостатньо вивчені‚ проте на Заході вже пропонують спеціальні курси лікування від таких профзахворювань. Проводячи довгий час за монітором маємо перш за все проблеми із зором. Причини її виникнення: коли ми уважно дивимося на монітор, то моргаємо в 3 рази рідше, ніж в звичному стані. Відблиски і віддзеркалення в моніторі створюють додаткову напругу. Прояви: паління, світлобоязнь, свербіння, почервоніння, швидка стомлюваність, відчуття «піску» в очах‚ головна біль.

Стан сітківки покращує вітамін А‚ перерви між роботою на комп'ютері і профілактична серія моргань кожні 15 хвилин. Кожні півгодини бажано відвертатися від монітора і дивитися у далечінь. Окуляри з покриттям антивідблиску, якщо екран не рідкокристалічний. Монітор повинен знаходитися трохи нижче за рівень очей. Відстань від очей до монітора не повинна бути менша за 70 сантиметрів: ви повинні ледь діставати до екрану‚ витягнувши вперед руку. Правильне освітлення пом'якшує мигтіння на екрані.

Ще один недолік роботи з комп'ютером - нарікання на біль у лікті. Причини: працюючи за комп'ютером, людина спирається ліктьовою поверхнею на край столу. Через постійне тривале механічне роздратування виникає бурсит - запалення синовіальних сумок. Прояви: лікоть припухає, «горить», червоніє, з'являються болі; може підвищитися температура; функція суглоба помірно обмежена, але збережена, що відрізняє бурсит від артриту.

Стежте за тим, як розташовуються ваші руки під час роботи за комп'ютером, не спирайтеся ліктьовою поверхнею на край столу. Кладіть руки повністю на стіл або тримайте в піднятому стані. Якщо в робочому кріслі є підлокітники‚ вони мають відповідати зросту і впиратися в лікті‚ оскільки вам доведеться дуже високо підіймати плечі‚ а це защемлює шийні хребці.

Отже, в компанії «ТімТек» за весь час її існування не було зафіксовано жодного випадку виробничого травматизму чи нещасного випадку, а також жоден співробітник не постраждав від професійних захворювань, тому жодних економічних втрат підприємство не зазнало. А це в свою чергу свідчить про те, що компанія робить все для того, щоб запобігти нещасним випадкам або профзахворюванням.

**ВИСНОВКИ**

В результаті роботи над дипломною роботою бакалавра було проведено розробку програмного продукту для ведення обліку та грошових транзакцій підприємства, в середовищі візуальної розробки Embarcadero Delphi.

Також було досліджено існуючі рішення програмного забезпечення для ведення обліку на підприємствах.

Спроектовано базу даних MySQL, яка містить 10 таблиць. До 8 з них проведено зв’язки для об’єднання інформації різних таблиць.

В програмі розроблено функцію авторизації для обмеження прав доступу до програми. Розроблені фільтри пошуку даних по різних критеріях, в залежності від вмісту інформації в таблицях. Програма автоматично обчислює загальний прихід коштів, прихід коштів за місяць, регулярні витрати підприємства та не регулярні витрати за місяць. На основі цих обчислень нараховує зарплату працівникам.

Застосування такої програми дозволить автоматизувати облік підприємства та фінансові операції. Значно спростить роботу працівникам, які цим займаються, що збільшить діяльність підприємства в цілому.

**СПИСОК ПОСИЛАНЬ НА ДЖЕРЕЛА**

1. Вовк Р.Б., Шекета В.І. Методичні вказівки до виконання дипломної роботи бакалавра – Івано-Франківськ: ІФНТУНГ, 2013. - 42с.
2. <http://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BB%D1%96%D0%BA>
3. <http://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B0%D0%BB%D1%96%D1%82%D0%B8%D1%87%D0%BD%D0%B8%D0%B9_%D0%BE%D0%B1%D0%BB%D1%96%D0%BA>
4. <http://uk.wikipedia.org/wiki/Embarcadero_Delphi>
5. <http://document.ua/pravila-ohoroni-praci-pid-chas-ekspluataciyi-elektronno-obch-nor17970.html>
6. <http://uk.wikipedia.org/wiki/MySQL>
7. <http://uk.wikipedia.org/wiki/InnoDB>
8. <http://www.mysql.ru/docs/man/INSERT.html>
9. <http://www.mysql.ru/docs/man/SELECT.html>
10. <http://www.mysql.ru/docs/man/UPDATE.html>
11. <http://www.mysql.ru/docs/man/DELETE.html>
12. <http://www.mysql.ru/docs/man/JOIN.html>
13. <http://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BA%D0%B5%D1%80%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85>
14. <http://ua.textreferat.com/referat-2750.html>
15. <http://knowledge.allbest.ru/audit/2c0b65625b3bc68b4c43b89421206c27_0.html>
16. http://www.terinvest.com.ua/index.php?id=1090&show=9778