МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ТКАЧУК Оксана Олексіївна

Програмна система обліку споживання води тепломережею/ Software system for accounting the water consumption by heating network

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконала студентка групи
ПЗС-42
О. О. Ткачук

Науковий керівник:
к.е.н., доцент АВГУСТИН Р.Р.

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

ЗМІСТ

ВСТУП

В сучасних умовах спеціалізовані організації проводять воду підготовку і подачу води на підприємства і в будинку. З такими підприємствами необхідно розраховуватися за спожитий обсяг води, поставленої на договірних умовах.

Система обліку води дозволяє вести облік витрат води, збирати інформацію про споживання, на підставі якої надалі виробляються фінансові розрахунки за спожиту воду. На підставі даних, отриманих від системи обліку води, можна впроваджувати заходи по заощадженню питної води.

Системи обліку води встановлюються в муніципальних і приватних домоволодіннях, компаніях, які керують житловими фондами, а також в компаніях-постачальниках води.

Система обліку визначається тим, що після виконання вимірювань (зняття контрольних показань) їх результати для входження в систему балансу (міста, локальної зони, водопровідної станції) піддаються математичній і логічній обробці, що забезпечує:

- облік і обробку позаштатних ситуацій;
- виключення недостовірних результатів вимірювань;
- формування часових, добових або місячних архівів.

Впровадження автоматизованої системи обліку витрати води дозволить виробляти автоматичний збір, накопичення, обробку, зберігання і відображення отриманої інформації, а також її передачу по дротових або бездротових каналах зв'язку на центральний диспетчерський пункт водопостачальної організації.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПІДТРИМКИ ПРОЦЕСІВ ОБЛІКУ СПОЖИВАННЯ ВОДИ

1.1. Коротка характеристика об’єкту управління

Метою діяльності комунального підприємства “Тернопільводоканал” є забезпечення стабільності роботи систем водопостачання і водовідведення та зниження виробничих витрат. Підприємство планує досягнення цієї мети шляхом подальшого удосконалення роботи по таких напрямках як бухгалтерський облік та фінансовий менеджмент, формування тарифів, експлуатація та технічне обслуговування основних засобів, нарахування плати за послуги та збирання платежів від споживачів, залучення громадськості до процесу прийняття стратегічних рішень. Для технічного переоснащення й оновлення водопровідно-каналізаційних об’єктів комунальної інфраструктури підприємство потребує збільшення обсягів капітальних інвестицій.

Комунальне підприємство “Тернопільводоканал”, надалі “Підприємство”, засноване на комунальній власності і безпосередньо підпорядковане управлінню житлово-комунального господарства Тернопільської міської Ради, у своїй діяльності керується Законом України “Про підприємство”, та іншими нормативними актами з врахуванням особливостей, передбачених даним Статутом .

Підприємство набуває статусу юридичної особи з моменту його державної реєстрації, має самостійний баланс і статутний фонд, розрахунковий та інші рахунки в установах банків, круглу печатку, кутовий штамп, фірмовий бланк зі своїм найменуванням і інші реквізити. Підприємство може від свого імені укладати на всій території України і за межами угоди, інші акти з самостійними суб’єктами підприємницької діяльності.

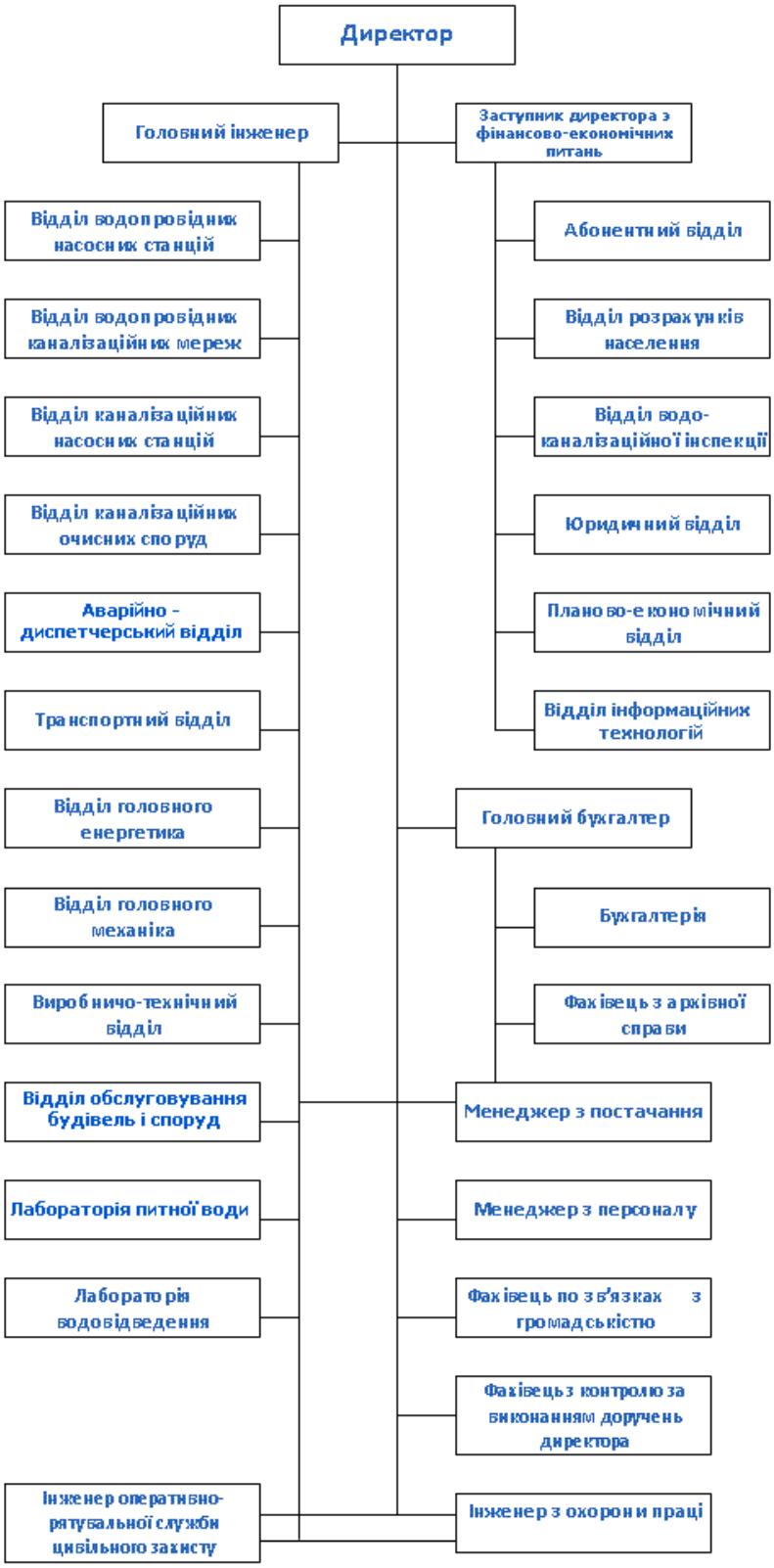

Рис. 1.1. Структура КП “Тернопільводоканал”

Підприємство погоджує з міською Радою і іншими установами заходи, які викликають екологічні, соціальні, демографічні зміни і несе матеріальну відповідальність за шкідливі наслідки своєї діяльності.

Підприємство набуває майнові та немайнові права і обов'язки, виступає позивачем і відповідачем в суді, арбітражному, третейському судах, відповідає по своїх зобов'язаннях усім належним йому майном.

Власником Підприємства є Тернопільська міська Рада народних депутатів. Засновником підприємства є виконавчий комітет Тернопільської міської Ради. Власник не відповідає за борги і зобов'язання Підприємства, а Підприємство не відповідає за зобов'язаннями власника засновника.

Підприємство створене з метою проведення робіт і надання послуг по водопостачанню юридичних і фізичних осіб в м. Тернополі, забезпечення належного технічного стану водопровідно-каналізаційних мереж і очисних споруд міста.

Предметом і метою діяльності підприємства є:

- водопостачання м. Тернополя шляхом експлуатації артезіанських свердловин і поверхневих джерел водопостачання;
- здійснювання робіт по реалізації води, прийманню і очистці стоків;
- надання послуг населенню по подачі води і відведенню стоків;
- забезпечення сталої роботи мереж водопроводу і каналізації шляхом експлуатації підземних і наземних споруд;
- вирішення питань реконструкції і розширення водопровідних і каналізаційних споруд і сіток в поточному році і на перспективу;
- здійснення встановлення, ремонту та заміни засобів обліку споживання води.

Підприємство проводить заходи по поліпшенню виробничих процесів, впровадженню у виробництво нової техніки, технології, підвищенню продуктивності праці, зниженню норм використання сировини, матеріалів, паливно-енергетичних ресурсів, забезпеченню матеріально-технічної бази потрібними матеріалами, виробничими і складськими приміщеннями.

Підприємство здійснює свої права в питаннях планування, капітального ремонту, удосконалення техніки, технології виробництва, матеріально-технічного постачання та збуту, фінансів, праці, заробітної плати, а також в інших питаннях своєї діяльності.

Підприємство здійснює свою діяльність за виробничо-господарським планом. Підприємство здійснює комерційну і зовнішньоекономічну діяльність. Підприємство здійснює автотранспортом перевезення вантажу і пасажирів в межах України і країн СНД.

Управління Підприємством здійснюється на основі поєднання прав засновника щодо господарського використання свого майна і самоврядування трудового колективу. Органи управління Підприємства: загальні збори трудового колективу, директор.

Аналіз обігу та використання трудових ресурсів. Станом на 01.01.2015 року загальна чисельність працюючих на підприємстві становила 641 осіб, серед них:

- керівники – 22 особи (3%);
- спеціалісти та службовці – 98 осіб (15%);
- робітники – 521 особа (82%).

Таблиця 1.1

Структура і динаміка кадрового складу підприємства.

Категорія персоналу	Кількість осіб	% до підсумку
Виробничий персонал	351	55
Загально-виробничий персонал	168	26
Адміністративний персонал	31	5
Персонал служби збуту	77	12
Інші працівники	14	2
Всього працюючих	641	100

Середній вік працюючих на підприємстві складає 44,6 роки, а частка осіб пенсійного та перед пенсійного віку – 17,2% від загальної кількості персоналу (109 осіб).

Адміністрація підприємства коригує чисельність персоналу відповідно до фактичних потреб. Так, нормативна чисельність працівників на підприємстві станом на 01.01.2015 р. складає 1007 осіб, фактично працює - 641 особи, що дозволяє заощадити фонд оплати праці. Підприємство оплачує робітникам частину фонду заробітної плати за розширення зони обслуговування, суміщення професій. Частка витрат на оплату праці складає 15% від витрат звичайної діяльності.

Через відносно низький рівень заробітної плати важко залучати та утримувати висококваліфікованих спеціалістів. Крім цього, на підприємстві спостерігається плинність молодих кадрів, що пов’язано з важкими умовами праці та низьким рівнем заробітної плати, особливо на дільницях водо-, каналізаційних мереж, де роботи виконуються за будь-якої погоди.

Служба по роботі зі споживачами КП “Тернопільводоканал” складається із абонентського відділу, відділу розрахунків населення, водо-, каналізаційної інспекції та юридичного відділу.

Розрахунки із споживачами – юридичними особами здійснює абонентський відділ. Спеціалісти відділу укладають договори на подачу холодної води та водовідведення юридичним особам, реєструють акти прийняття приладів обліку, приймають і обробляють показники приладів обліку води від споживачів, складають акти по фактично наданим послугам водопостачання, відстежують заборгованість юридичних осіб, здійснюють зведення місячної звітності за надані послуги відповідно до показників приладів обліку.

Відділ розрахунків населення здійснює облік споживання, нарахувань та розрахунків за спожиті послуги, надані пільги та субсидії. В штаті відділу працюють 53 спеціалісти. Відділ складається із розрахункової групи та групи контролю.

Спеціалісти розрахункової групи виконують нарахування за надані споживачам послуги, зняття показників будинкових лічильників та облік фактичного споживання послуг, облік платежів та заборгованості, оформлення

та супровід договорів на надання послуг населенню, оформлення та супровід договорів реструктуризації заборгованості за надані послуги.

До обов'язків розрахункової групи входить також виконання наступної роботи, пов'язаної з обслуговуванням споживачів: формування та видача довідок для оформлення субсидій, реєстрація пільг, внесення змін до складу сім'ї (за даними паспортного столу).

Група контролю здійснює рознесення по 20 дільницях рахунків за послуги з водопостачання та водовідведення (до 13-14 числа кожного місяця), попереджень про наявність заборгованості за надані послуги по поштових скриньках, досудових попереджень боржникам, які мають критичну заборгованість, виконує контроль виконання договорів реструктуризації боргу. До обов'язків групи контролю входить робота із власниками квартирних лічильників – зняття показників, технічне обстеження зі складанням відповідних актів за заявками мешканців або за дорученням департаменту ЖКГ міської ради (в будинках, де зафіксоване значне зростання фактичного споживання).

Водо-, каналізаційна інспекція виконує технічні заходи обслуговування споживачів:

- робота з боржниками із застосуванням технічних засобів впливу;
- відключення споживачів-боржників;
- опломбування встановлених лічильників;
- контроль за підключенням до мереж нових споживачів (санкціонованих, несанкціонованих);
- обслуговування внутрішньобудинкових мереж в неробочі години та у вихідні або святкові дні (на випадок виникнення аварійних ситуацій);
- робота з мешканцями сіл, що користуються водопостачанням на шляху водопроводу;

Юридичний відділ забезпечує правовий супровід роботи із споживачами, укладання договорів на надання послуг, проведення заходів по відшкодуванню

боргів, реструктуризації заборгованості, судове провадження позовів до найбільших боржників.

Стимулювання праці. На підприємстві запроваджено систему преміювання контролерів відділу розрахунків населення, згідно з якою до окладів застосовуються коефіцієнти за рознесення рахунків споживачам та за збір показників лічильників.

1.2. Опис предметної області

Водопостачання - одна з найважливіших галузей техніки, спрямована на підвищення рівня життя людей, благоустрій населених пунктів, розвиток промисловості та сільського господарства.

Водопостачання базується на використанні природної сировини - води, запаси якої, як і інших природних ресурсів, обмежені. Це зумовлює необхідність розумного і дбайливого ставлення до води.

Під водопостачанням розуміють сукупність заходів щодо забезпечення водою різних її споживачів.

Системою водопостачання (водопроводом) називається комплекс інженерних споруд і пристроїв, які здійснюють такі завдання: забір води з природних джерел, поліпшення показників її якості до заданих норм, транспортування на необхідні відстані, зберігання її запасів, подача і розподіл споживачам.

Під системою водопостачання також може матися на увазі комплекс взаємопов’язаних споруд, призначених для водозабезпечення будь-якого об’єкта або групи об’єктів. Система водопостачання, що забезпечує водою окремі райони або групи населених пунктів, або групи промислових об’єктів, називається районної або груповий системою водопостачання.

Всі сучасні системи водопостачання населених місць є централізованими: кожна з них забезпечує водою велику групу споживачів.

Централізована система водопостачання населеного пункту або промислового підприємства повинна забезпечувати прийом води з джерела, її кондиціонування (якщо це необхідно), транспортування і подачу до всіх споживачів під необхідним тиском. З цією метою в систему водопостачання повинні бути включені: водоприймальні споруди, призначені для отримання води з природних джерел; насосні станції, що створюють напір для передачі води на очисні споруди, в акумулюючі ємності або споживачам; споруди для обробки води; резервуари і водонапірні башти, які є запасними і регулюючими ємностями; водоводи і водорозподільні мережі, призначені для передачі води до місць її розподілу і споживання.

Для цілей водопостачання використовуються природні джерела води: поверхневі - відкриті водойми (річки, водосховища, озера, моря) і підземні (грунтові та артезіанські води і джерела). Для отримання води з природних джерел, її очищення відповідно до потреб споживачів і для подачі до місць споживання служать наступні споруди: водоприймальні споруди, насосні станції першого підйому, що подають воду до місць її очищення; очисні споруди; збірні резервуари чистої води; насосні станції другого або наступних підйомів, що подають очищену воду в місто або на промислові підприємства; водоводи і водопровідні мережі, службовці для подачі води споживачам.

Загальна схема водопостачання може видозмінюватися залежно від конкретних умов. Послідовність розташування окремих споруд системи водопостачання та їх складу можуть бути різними в залежності від призначення, місцевих природних умов, вимог водоспоживачів або виходячи з економічних міркувань.

Так, регулююча ємність може бути розташована в різних точках території об’єкта в залежності від поєднання планування об’єкта і рельєфу місцевості.

Проектування, будівництво та експлуатація системи водопостачання повинні, не порушуючи сформованого екологічної рівноваги навколишнього природного середовища (гідро- і біосфери), задовольняти вимогам надійності.

Різні джерела класифікують системи водопостачання з таких підстав:

- За характером вододжерела - з використанням поверхневих вод (річок, озер, водосховищ, морів); з використанням підземних вод; змішані;

- За способом подачі води - нагнітальні; гравітаційні; комбіновані;

- За призначенням - господарсько-питні; виробничі; протипожежні; об'єднані, що задовольняють потреби перерахованих споживачів в будь-якому поєднанні;

- За видами об'єктів, що обслуговуються - міські і селищні, промислові, колгоспні і радгоспні, залізничні та ін .;

- За територіальним охопленням водоспоживачів - місцеві (локальні), що забезпечують водою окремі об'єкти, промислові підприємства, залізничні станції, тваринницькі ферми; централізовані, що забезпечують водою всіх водоспоживачів даного міста або населеного пункту; групові або районні, службовці для забезпечення водою кількох населених пунктів у великому районі;

- За характером використання води - прямоточні, в яких воду після одноразового використання очищають і скидають в водойми; оборотні, в яких воду після використання для технічних цілей очищають і охолоджують, потім багаторазово споживають на тому ж об'єкті; з повторним використанням води;

- По надійності - однієї з трьох категорії в залежності від виду промислового підприємства, числа жителів в населеному пункті і вимог безперебійності подачі води (СПВ).

Системи водопостачання повинні надійно забезпечувати всіх споживачів водою належної якості в заданому кількості і під необхідним напором при найменших витратах на будівництво і експлуатацію споруд. При будівництві водопровідних споруд слід максимально використовувати індустріальні елементи, а при їх експлуатації широко застосовувати механізацію, автоматизацію і телемеханіку.

Проектування будь-якого водопроводу починається з вибору схеми, яка представляє собою сукупність споруд водопроводу і послідовність розташування їх на місцевості.

Факторами, що визначають вид схеми водопостачання, є: тип використовуваного джерела і якість води в ньому, вимоги, що пред'являються до води споживачами, рельєф місцевості, розміщення споживачів на плані, розміри водоспоживання, наявність природних і штучних перешкод зведенню водопровідних споруд, потужність вододжерела і його віддаленість.

Зазвичай в початковій стадії проектування складають два (або більше) можливі варіанти схем водопостачання. Після техніко-економічного розрахунку кожного варіанта їх порівнюють і вибирають найкращий. За обраною схемою остаточно проектують і розраховують всі пристрої системи водопостачання.

Характерними вимогами для виробничого водопостачання є його надійність в відношенні як сталості (і безперебійності) водоподачі, так і постійних напорів.

Великі споживачі технічної води (виробничі підприємства, громадські туалети) мають свої власні підключення до мережі технічної води. Підприємства харчової промисловості (хлібзаводи, молочарні, консервні заводи), а також підприємства, які споживають воду підвищеної якості, лікарні, поліклініки, амбулаторії, ветеринарні лікарні, аптеки можуть отримувати воду з питного водопроводу.

Під схемою водопостачання розуміють генеральний план об'єкта водопостачання із зазначеними на ньому водопровідними спорудами. Схеми водопостачання проектують на основі генеральних планів міст (перша черга - на термін 8-10 років і перспектива - на термін 20-25 років) і промислових підприємств.

Схема водопостачання залежить від багатьох факторів, у тому числі головними є наступні: місце розташування, потужність і якість води джерела водопостачання, рельєф місцевості і кратність використання води на промислових підприємствах.

Джерелом водопостачання можуть служити поверхневі водойми (ріки, озера, моря) і підземні води.

В даний час наука і техніка має все необхідне для вирішення складних проблем водопостачання і каналізації міст і промислових підприємств, а також для охорони водойм від забруднень.

Споживання води в містах і на промислових підприємствах протягом доби нерівномірне. У містах в нічний час води споживається значно менше, ніж днем. На промислових підприємствах на початку і кінці змін води для виробничих цілей витрачається менше, ніж в середині змін.

У містах і на промислових підприємствах витрачають велику кількість води. Її використовують на господарсько-питні та виробничі потреби, а також для пожежогасіння.

Забезпечення населення водою питної якості підвищує рівень благоустрою міст, покращує їх санітарний стан і оберігає людей від різних епідемічних захворювань, що поширюються через воду.

Інтенсивний розвиток промисловості з кожним роком призводить до збільшення загальної кількості води, використовуваної для виробничих цілей. В даний час воно вже значно перевищує загальну кількість води, використовуваної на господарсько-питні потреби.

У промисловості воду використовують як сировину при виготовленні продукції, середовища, в якій протікають технологічні процеси, а також для миття сировини, охолодження обладнання та інших цілей. У багатьох випадках вода знаходиться в безпосередньому контакті з сировиною або продукцією. Якість води та організація постачання підприємств водою впливають на остаточну якість і собівартість продукції.

Для пожежогасіння в містах і на промислових підприємствах воду використовують порівняно рідко і протягом короткого часу, але в більших кількостях.

Як вже зазначалося нами вище, для забезпечення міст і промислових підприємств водою будують системи водопостачання - комплекс інженерних споруд, а також заходів, що забезпечують отримання води з природних джерел, її очищення, транспортування і подачу споживачам.

Водопровідна вода в процесі використання в господарських, виробничих та інших цілях забруднюється і змінює свої властивості. Таку воду називають стічної. Стічні води, що утворюються в містах і на ряді промислових підприємств, містять органічні забруднення, які здатні загнивати і можуть служити середовищем для розвитку різних мікроорганізмів, у тому числі патогенних (хвороботворних). Стічні води багатьох підприємств містять шкідливі мінеральні домішки, хімічні сполуки або токсичні речовини.

Очисні споруди можна розміщувати поблизу як водоприймачів, так і об’єктів водопостачання.

Водовідведення - це комплекс інженерних споруд і санітарних заходів забезпечують прийом стічних вод від населення і промислових підприємств, транспортування і очищення їх з подальшим скидом в річку або на рельєф.

Для створення сприятливих санітарних умов на територіях міст і промислових підприємств стічні води слід видаляти за їх межі, а для виключення забруднення водойм стічні води потрібно очищати і знезаражувати. Для цього використовують системи каналізації. Каналізація - це комплекс інженерних споруд, що забезпечують збір стічних вод, транспортування їх за межі територій міст і промислових підприємств, а також їх очищення і знезараження.

Перш в містах використовували так звану вивізне каналізацію. Покидьки, розбавлені водою, збирали в спеціальні ємності (вигреби) і періодично вивозили автомобільним транспортом на спеціально відведені площі землі - асенізаційні поля.

Більш досконалою є сплавна каналізація, мережа підземних трубопроводів, по яких стічні води видаляються самопливом. У разі необхідності вони перекачуються в водойму або на очисні споруди, де піддаються інтенсивної очищення та знезараження.

Сплавна каналізація дає можливість здійснити належне водопостачання міст і промислових підприємств і створити сучасні впорядковані міста з

великою щільністю населення (з забудовою будинками великої поверховості) та вельми сприятливими санітарними умовами.

Очисні споруди обробляють природну воду з метою надання їй якостей, що відповідають вимогам споживачів. Очищена вода подається до об’єкта по водоводах і розлучається з його території водопровідною мережею. До вуличної мережі приєднуються будинкові відгалуження, по яких вода вводиться в будівлі. Всередині будівель влаштовується мережа внутрішнього водопроводу, що підводить воду до точок її розбору через різні водорозбірні пристрої (крани).

Стічні води здатні порушити санітарно-епідеміологічне благополуччя населення міст і промислових підприємств. Вони є джерелом забруднення навколишнього природного середовища. Системи водовідведення усувають негативні наслідки від впливу стічних вод на навколишнє природне середовище. Після очищення стічні води зазвичай скидаються у водойми.

Водовідведення здійснюється за допомогою комплексу підземних самопливних трубопроводів, очисних та інших споруд, за допомогою яких здійснюється відведення використаних і відпрацьованих вод, очищення та знезараження їх, а також обробка і знешкодження утворюються при цьому опадів з одночасною утилізацією цінних речовин. Такі комплекси називаються системами водовідведення, або водовідведенням.

Системи водовідведення усувають негативні наслідки від впливу стічних вод на навколишнє природне середовище. Після очищення стічні води зазвичай скидаються у водойми.

Найбільш досконалими системами водовідведення є такі, які забезпечують очистку та підготовку води такої якості, при якому можливе повернення води для повторного використання в промисловості або сільському господарстві. Такі системи називаються безстічними або замкнутими.

1.3. Огляд і аналіз існуючих аналогів, що реалізують функції предметної області

З метою проектування якісної інформаційної системи обліку споживання води необхідно провести аналіз відомих систем, які вирішують або частоко вирішують поставлену задачу. Необхідно виділити їх основні переваги та недоліки.

Система дистанційного обліку та контролю споживання води компанії “Sea electronics”

Функції, виконувані системою:

• автоматичний збір інформації з приладів обліку;

• читання миттєвих показань приладу обліку за запитом оператора;

• накопичення в терміналі інформації по споживанню на початок періодів (на початок доби, на початок місяця);

• накопичення в терміналі інформації по потужності споживання за 30 хвилинний період інтеграції, для аналізу аварійних ситуацій і виявлення розкрадання;

• автоматична передача накопичених даних в диспетчерський центр і запис інформації, що надходить в базу даних за хронологією отримання, в ретроспективі;

• візуалізація накопичених даних у вигляді таблиць і графіків;

• дистанційна передача звітів по електронній пошті;

• аналіз сигналів з додаткових датчиків (охорона пункту обліку, реєстрація спроб впливу на систему).

Система складається з диспетчерського центру і терміналів збору даних.

Програмне забезпечення диспетчерського центру дозволяє приймати і накопичувати в базі даних показання лічильників і звіти про події, що відбуваються на пунктах обліку (провалля харчування, розтин об'єкта і т.п.). При кожному сеансі обміну між терміналом і диспетчерським центром передаються пакети інформації, що містять:

- поточні показання приладу обліку, показання приладу обліку на кожен день, в ретроспективі, за минулі 31 день;

- показання на перший день місяця, в ретроспективі, за минулі 12 місяців, а також 30-ти хвилинні профілі потужності споживання.

Ці дані візуалізуються у вигляді таблиць і у вигляді графіків. Диспетчер має можливість детально вивчити дані і виявити випадки викрадення або аварійні ситуації на мережах споживання.

За накопиченим даними будуються звіти - групові та індивідуальні, наприклад, для груп терміналів - стан на конкретну дату, і індивідуально, розгорнутий, по кожному терміналу, з відображенням півгодинних потужностей і показань приладу обліку на певні дати. Звіти можуть розсилатися адресатам по електронній пошті.

Апаратна частина терміналу збору даних є спеціалізованою плату, на якій встановлений GSM модуль, джерело живлення і вхідні кола сполучення з імпульсними входами лічильників.

Мікроконтролер на базі скрипта здійснює підрахунок імпульсів на цифрових входах і запис отриманих відліків в пам'ять. При відсутності електроживлення модуль продовжує працювати в режимі наднизького енергоспоживання, при цьому триває підрахунок імпульсів, GSM частина відключається і мікропроцесорна частина працює автономно.

При подачі зовнішнього живлення модуль автоматично запускає GSM частина і переходить в повну готовність. Для підвищення надійності мікроконтролер стежить за функціональним станом GSM частини і виробляє перезапуск її, в разі необхідності. GSM частина працює під управлінням OpenCPU додатки, яке дозволяє організовувати канал передачі даних з використанням всіх технологій доступних в GSM - таких як GPRS, CSD, SMS а також DTMF посилок в голосовому каналі.

Корпус терміналу виготовлений із пластику, має гермовводи для вводу кабелів (харчування, інформаційного і антенного).

Економічний ефект забезпечується:

• відсутністю людського фактора при знятті показань працівниками-контролерами;

• дистанційним зчитуванням приладів обліку, що дозволяє скоротити робочий час і транспортні витрати, необхідні при об'їздах для зняття показань;

• можливістю виявляти зміни споживання, що дозволяє оперативно виявляти аварійні ситуації або випадки розкрадання (тобто навмисні дії, спрямовані на отримання обсягів холодної води в обхід лічильника);

• високою надійністю, забезпеченої незалежної двухпроцессорной системою;

• гнучкістю системи, наявністю шин розширення, що дозволяють проводити модернізацію, без заміни терміналу, при розширенні функціоналу системи.

На рисунку 1.2 представлено інтерфейс розглянутої вище системи.

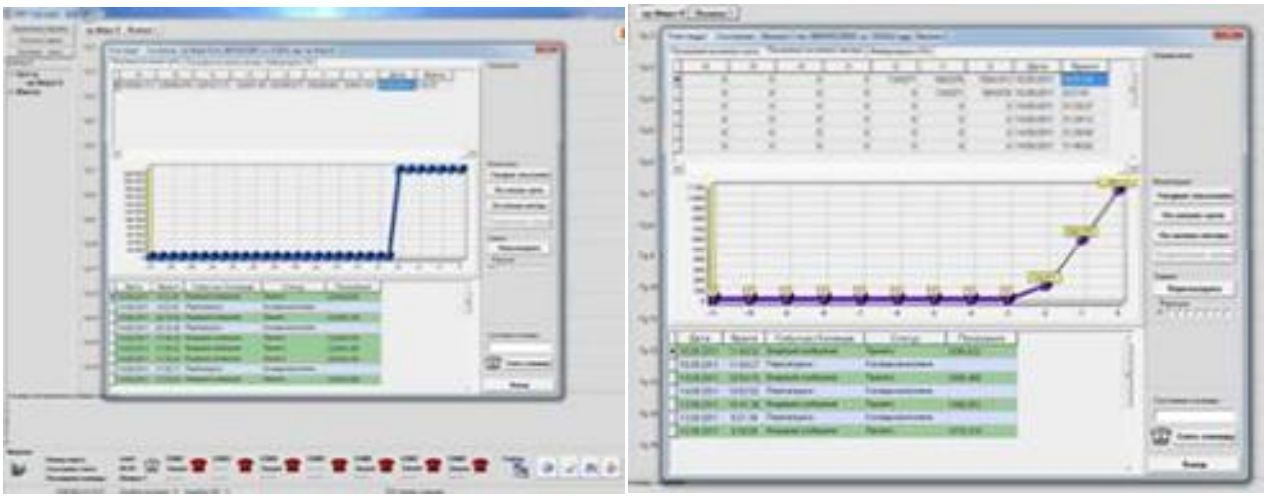

Рис. 1.2. Інтерфейс програми обліку та контролю споживання води компанії "SEA Electronics"

Основні відмінності системи:

• використання бездротового GSM мікропроцесора з вмонтованим спеціалізованим програмним забезпеченням, що виконує всі функціональні завдання з передачі даних;

• використання мікроконтролера з наднизьким енергоспоживанням, яка провадить підрахунок імпульсів з приладу обліку, незалежно від наявності зовнішнього електроживлення.

Автоматизована система комунального обліку витрат води "Есіон" . Автоматичне зняття показань квартирних лічильників води і передача знятих показань в центр обробки (керуючу компанію, ТСЖ) з подальшим узагальненням результатів і поданням їх у вигляді рахунку-квитанції на оплату комунальних послуг.

Основні складові системи:

- квартирний вузол обліку води;
- лічильники з імпульсними виходами;
- прилад збору і бездротової передачі даних;
- система ретрансляторів передачі даних;
- концентратор потоків даних;
- АРМ оператора.

Загальний принцип функціонування. Водолічильник виробляє послідовність імпульсів, які сприймаються і підсумовуються приладом збору і бездротової передачі даних.

Залежно від настройки системи (на вимогу оператора або через заданий період) прилад транслює поточні показання водолічильників в концентратор потоків даних. Останній їх конвертує і передає в АРМ оператора.

Передача даних від квартирного вузла обліку до концентратора є бездротової. Для підвищення надійності системи передачі може використовуватися система ретрансляторів, установлюваних в слабкострумових поверхових щитках.

Остаточна обробка інформації про споживаних ресурсах здійснюється засобами АРМ оператора, включаючи підготовку рахунків-квитанцій на оплату комунальних послуг.

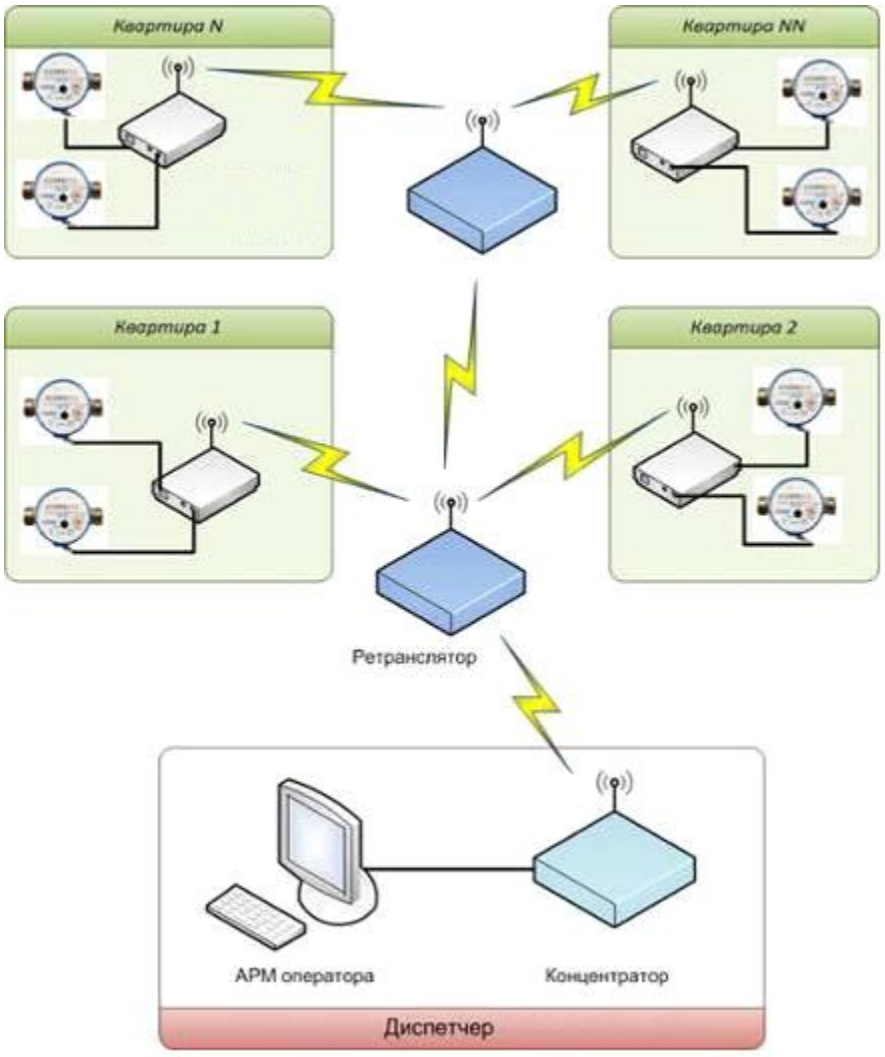

Рис. 1.3. Схема роботи автоматизованої системи комунального обліку витрати води "ЕСІОН"

Ефект від впровадження:

• Об'єктивність знімання показників (запобігання підробленю показників водолічильників мешканцями).

• Оперативність і одночасність зняття показань (точна синхронізація показань квартирних водолічильників і загальнобудинкового вузла обліку).

• Забезпечення можливості об'єктивного автоматизованого аналізу витрат води (виявлення несанкціонованого відбору води, аварійних протікань, нераціонального використання води на загальнобудинкові потреби та ін.).

Прогнозування споживання води домоволодінням (забезпечує оптимальний вибір періодів регламентних, профілактичних, планових ремонтних робіт на мережах, оптимізація режимів роботи котелень).

Вимірювальна автоматизована система контролю та обліку витрат води "Пульсар"

Призначення - автоматизований комерційний і технологічний облік споживання холодної води.

Склад системи:

• лічильники енергоресурсів, внесені в Держреєстр засобів вимірювань України, оснащені імпульсним телеметричним виходом або цифровим виходом;

• лічильники імпульсів - реєстратори "Пульсар" - вторинні прилади, до кожного з яких підключаються до шістнадцяти первинних лічильників з імпульсним виходом. Використовуються для накопичення числоїмпульснимі інформації з первинних лічильників з прив'язкою її до астрономічного часу, передачі даних в цифровому форматі на комп'ютер диспетчера (стандарт RS485);

• пристрої збору і передачі даних, що забезпечують збір даних з реєстраторів "Пульсар", з лічильників енергоресурсів з цифровим виходом, зберігання і передачу даних на верхній рівень системи, синхронізацію роботи приладів обліку. Пристрої встановлюються безпосередньо на об’єкті. Використання пристроїв не є обов'язковою умовою роботи системи;

• допоміжні пристрої, що забезпечують передачу цифрової інформації (перетворювачі, ретранслятори, модеми, блоки живлення);

• сервер комерційного обліку, автоматизовані робочі місця.

Функції системи:

• ведення бази даних споживання ресурсів на ПК;

• підготовка аналітичної інформації, звітів, протоколів, графіків для подальшого друку;

• виписка рахунків абонентам для оплати спожитих ресурсів;

• інформування споживачів про стан оплати і споживанні ресурсів;

• зведення внутрішньооб'єктного балансу надходження і споживання ресурсів з метою виявлення вогнищ несанкціонованого споживання;

• видача даних і обмін аналітичною інформацією між структурами ЖКГ і водопостачальними організаціями;

• коригування внутрішнього годинника лічильників імпульсів - реєстраторів і лічильників ресурсів з цифровим виходом;

• контроль ліній зв'язку з лічильниками ресурсів;

• захист інформації від несанкціонованого доступу.

Переваги:

1. Доступна вартість устаткування і монтажу. Використовується мінімум функціональних блоків і мінімальна довжина проводів, що досягається шляхом використання паралельного принципу підключення лічильників імпульсів - реєстраторів до загальної лінії.

2. Надійність. Вся інформація про споживання ресурсів до її введення в ПК зберігається в незалежній пам'яті лічильників імпульсів - реєстраторів. У разі відключення живлення мережі, реєстрація даних продовжується. Відсутність проміжних блоків накопичення інформації між лічильником імпульсів - реєстратором і комп'ютером дозволяє мінімізувати ймовірність псування даних і виникнення збоїв в роботі системи. Використання апаратних засобів передачі даних за протоколом RS485 виключає вплив наведень, перешкод та ін. При передачі даних.

3. Зручність і простота обслуговування. Персоналу, налаштовує та обслуговує систему, не обов'язково спеціально проходити тривале навчання, мати відповідну освіту і т.д. Інтерфейс програмної частини, як і всієї структури системи, інтуїтивно зрозумілий і простий. Використання адаптера 485/232 дозволяє зчитувати інформацію в ПК прямо на місці. У разі наявної вільної телефонної лінії зручно передавати інформацію на віддалений комп'ютер через звичайний телефонний модем. У разі, якщо телефонна лінія відсутня, зручно передавати інформацію через GSM-модем. Оперативний контроль за роботою

головної функціональної осередки системи - лічильника імпульсів - реєстратора, можливий на місці за показниками вбудованого РКІ. Практично необмежені можливості по довжині лінії зв'язку і кількості лічильників - реєстраторів в мережі роблять систему універсальною для застосування на різних типах об'єктів.

4. Різноманітність функцій. Різноманіття функцій відповідає всім сучасним вимогам до подібних систем. Є можливість нарощування функцій без зміни загальної структури системи.

5. Відкритість, сумісність, захищеність. Система побудована на основі відкритих протоколів передачі даних, проте дані захищені від несанкціонованого зчитування. Система має власних OPC - сервер. Споживач інформації при роботі з даними може користуватися як програмним забезпеченням, поставленим разом з системою, так і власним програмним забезпеченням. Програмне забезпечення сумісне з розрахунковими програмами.

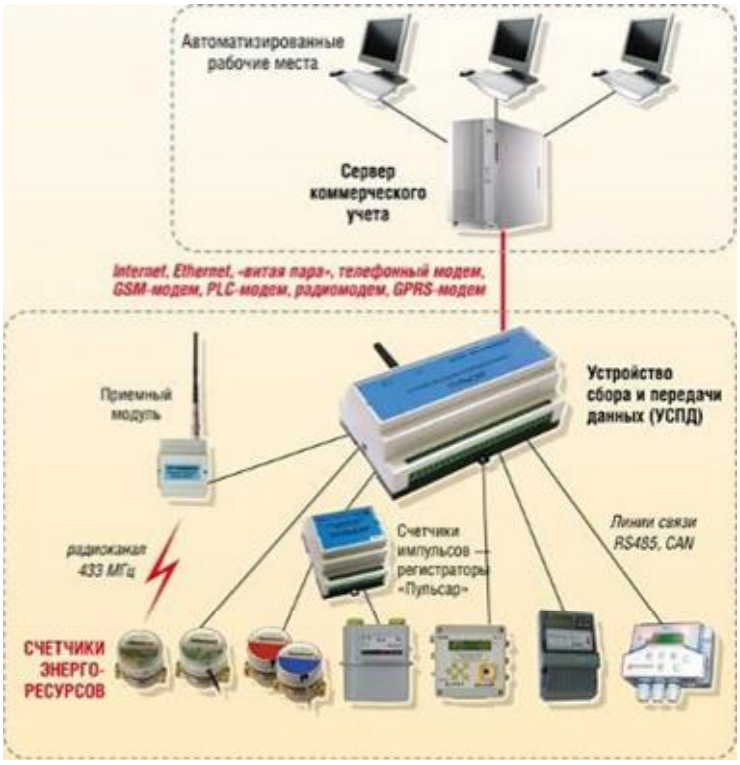

Рис. 1.4. Структура вимірювальної автоматизованої системи контролю та обліку витрати води "Пульсар"

Система комерційного поквартирного обліку води "Сатурн". Система обліку призначена для вимірювань, комерційного поквартирного обліку обсягу, витрати холодної води, моніторингу параметрів систем водопостачання, збору, зберігання, візуального представлення, документування результатів вимірювань та інформації про споживання води при комерційних розрахунках між споживачем і водопостачальної організацією.

Інформаційно-вимірювальної система комерційного поквартирного обліку води складається з автоматизованого робочого місця (АРМ) оператора системи, встановленого в диспетчерському пункті та обслуговуючого групу будинків (мікрорайон), інформаційної комп'ютерної мережі передачі даних з каналообразующей апаратурою (рекомендується використання вже існуючої мережі провайдера, наприклад, волоконно оптичної мережі, Wi-Fi радіоканалу), а також лічильників води, реєстраторів, блоків рахунку імпульсів БТС-2, БРК-К, встановлених в житлових будинках і перетворювачі інтерфейсів БПДД-RS, БПДД-CAN. Управління системою і контроль її роботи здійснюється з єдиного автоматизованого робочого місця АРМ оператора системи обліку на основі типового персонального комп'ютера. Інформація про спожитий обсяг води по кожній квартирі відображається на моніторі АРМ оператора у вигляді таблиць і графіків. Для документування статистичної інформації про роботу систем водопостачання, АРМ оператора оснащений принтером.

Знімання даних з квартирних лічильників холодної води здійснюється одним із двох способів:

- по провідних лініях зв'язку лічильники підключаються до блокам рахунку імпульсів БТС-2;

по радіоканалу 433 МГц - лічильники підключаються до квартирних БРК-К, які передають інформацію по радіоканалу в поверховий БРК-Е.

У першому випадку БТС-2 встановлено на поверсі і приймає дані від 4 квартир, а в другому випадку в кожній квартирі встановлюють по одному БРК-К і один БРК-Е.

Імпульсні сигнали пропорційні обсягу води, що формуються лічильниками води, надходять в блоки БТС-2 або БРК-К, які підсумовують кількість імпульсів по кожному каналу. БТС-2, БРК-К містять вбудований джерело живлення, яке забезпечує рахунок імпульсів протягом 6 років. Потім вимірювальна інформація по домовик провідний інформаційно-живильної лінії інтерфейсу СОС-95 зчитується майстер-пристроєм ВКД-МЕ з блоків БТС-2, БРК-Е. Далі вимірювальна інформація надходить по мережі в комп'ютер АРМ оператора, який з встановленою періодичністю зчитує дані від всіх лічильників води.

Всі дані про обсяги води надходять до бази даних системи комерційного обліку. База даних встановлена на тому ж комп'ютері. АРМ оператора забезпечує відображення обсягу холодної води, виміряного квартирними водолічильниками.

АРМ формує документовані звіти за параметрами споживання води на основі запиту до бази даних системи комерційного обліку. АРМ оператора формує довідку про фактичне споживання води по квартирних приладів обліку за заданий звітний період часу. Звіти являють собою спеціально сформовані документи, встановленої форми, що містять інформацію про споживання води квартирою по заданому адресою за заданий інтервал часу. Також АРМ здійснює експорт обробленої вимірювальної інформації в заданому форматі файлів в автоматизовану систему нарахування оплати за користування житлово-комунальними послугами.

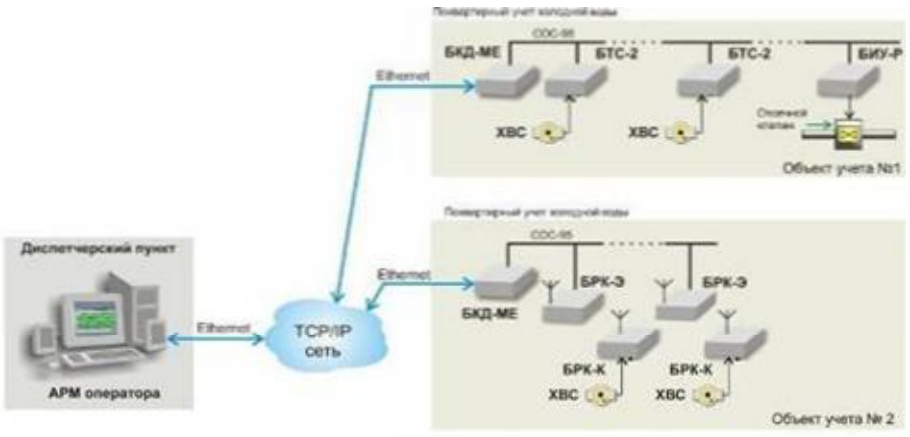

Рис. 1.5. Структура системи комерційного поквартирного обліку води "Сатурн"

Додатково АРМ оператора дозволяє в ручному режимі дистанційно відключати подачу води в квартиру за допомогою блоків управління Биу-Р, які підключені до відсічним клапанів подачі води. Відключення води відбувається під час вступу до Биу-Р команд з диспетчерського пункту. Биу-Р формує сигнал реле для відключення отсечного клапана води, встановленого на вводі в квартиру.

Система комерційного поквартирного обліку води може бути побудована в рамках окремого будинку, району, міста - в будь-яких місцях, де є можливість підключення до локальної або глобальної обчислювальної мережі. Можлива робота системи по виділених VPN-каналів через постачальників Інтернет-послуг.

З аналізу роботи діючої системи обліку витрати води, можна зробити висновки, що в їх роботі є ряд недоліків:

1. Не застосовуються витратоміри-лічильники, що формують архіви обсягів позаштатних ситуацій, тобто режимів роботи приладу в умовах експлуатації, що не відповідають технічній документації на прилад. Також не фіксуються вид і тривалість нештатної ситуації.

2. Застосовуються витратоміри-лічильники, які не запрограмовані на облік обсягу спожитої води під час позаштатних ситуацій.

3. Немає централізованого автоматизованого прийому сигналу і передачі даних в диспетчерську

Переваги системи-прототипу:

- простота системи;
- легкість ремонту при виникненні поломок приладів.

Недоліки системи-прототипу:

- немає достовірної картини даних при надзвичайних ситуаціях;
- не дозволяє виробляти централізований збір даних.

Переваги систем з огляду:

- сучасна елементна база дозволяє вести облік при виникненні надзвичайних ситуацій, а також виробляти дистанційне відключення водопостачання окремого споживача;

- дозволяє виробляти автоматизований збір і передачу даних на центральний сервер водопостачальної організації.

Недоліки систем з огляду: для обслуговування та ремонту компонентів і вузлів системи потрібно висококваліфікований обслуговуючий і ремонтний персонал.

1.4. Специфікація вимог до системи

Система призначена для автоматизованого обліку послуг водопостачання й водовідведення, зроблених населенню та підприємствам. За допомогою комплексу виконуються операції розрахунку нарахувань вартості послуг, обліку й контролю платежів, боргів як для підприємств так і для побутових споживачів, формує необхідні форми звітів.

Для користувача повинний бути передбачений унікальний авторизований вхід. У програмі повинний бути передбачений контроль збереження цілісності даних при аварійних ситуаціях у роботі програми. Вибір середовища розробки програми орієнтований на використання безкоштовних програмних засобів.

До принципових задач, які повинна вирішувати розроблювана програма, варто віднести наступні:

— програма повинна забезпечувати збереження та ефективний пошук усіх даних пов'язаних з абонентами та розрахунками з ними за надані послуги;

— програма повинна мати зручний інтерфейс;

— програма повинна підвищити швидкість обробки даних зв'язаних з розрахунками з абонентами;

— програма повинна мати у своєму розпорядженні засоби захисту від несанкціонованого доступу до даних, а також перегляду та їх модифікації другими програмними засобами;

— програма повинна знести до мінімуму можливість помилок у роботі з даними, які може зробити оператор;

— програма повинна передбачити контроль операцій (запит на підтвердження) які можуть привести до модифікації та пошкодження даних;

— програма повинна забезпечувати контроль даних що додаються у базу даних;

— програма повинна використовувати для своєї роботи операційну систему Windows.

До функціональних задач, які повинні вирішувати розроблювальна система, варто віднести наступні:

— збереження та пошук даних по прізвищу абонента;
— збереження та пошук даних по назві підприємства;
— збереження даних в довіднику міст;
— збереження даних в довіднику мікрорайонів;
— збереження та пошук даних в довіднику вулиць;
— збереження та пошук даних в довіднику будинків;
— збереження та пошук даних в довіднику лічильників;
— збереження та пошук даних в довіднику пільг;
— збереження та пошук даних в довіднику тарифів;
— збереження даних в довіднику видів платежів;
— збереження та пошук даних в довіднику штрафів;
— збереження та пошук даних в довіднику груп підприємств;
— збереження та пошук даних в довіднику норм для підприємств;
— збереження та пошук даних в довіднику тарифів для підприємств;
— збереження та пошук даних в журналі фізичних осіб;
— збереження та пошук даних в журналі юридичних осіб;

— збереження та пошук даних в журналі штрафів;

— збереження, пошук та фільтрація даних в журналі оплат фізичних осіб;

— збереження, пошук та фільтрація даних в журналі оплат юридичних осіб;

— збереження та пошук даних в журналі показань лічильників по холодній воді для фізичних осіб;

— збереження та пошук даних в журналі показань лічильників по гарячій воді для фізичних осіб;

— збереження та пошук даних в журналі показань лічильників по холодній воді для юридичних осіб;

— збереження та пошук даних в журналі показань лічильників по холодній воді для будинків;

— збереження та пошук даних в журналі показань лічильників по гарячій воді для будинків;

— можливість розрахунків та збереження їх результатів за надані послуги;

— формування звітів та рахунків для підприємств;

— формування рахунків для абонентів;

— захист бази даних від несанкціонованого доступу за допомогою пароля;

— забезпечення автоматичного розрахунку вартості наданих послуг;

— забезпечення автоматичного перерахунку при поверненнях;

— забезпечення автоматичного перерахунку при зміні тарифів;

— забезпечення автоматичного перерахунку при зміні пільг;

— забезпечення можливості зберігати звіти у файлах на жорсткому диску та їх завантаження;

— забезпечувати перегляд усіх можливих звітів та їх друк.

До сервісних задач, вирішувати які повинна розроблювана система, варто віднести наступні:

— програма повинна забезпечувати вибір зі списків даних там де це необхідно і можливо;

— програма повинна містити строку стану у якій було б можливо отримати коротку довідку по будь - якому елементу керування або редагування даних;

— програма повинна містити головне меню;

— програма повинна інформувати користувача про причину відмовлення у виконанні тієї чи іншої операції;

— програма повинна забезпечувати видалення записів по вказаному діапазону (наприклад за датою);

— запити, які виконує програма до бази даних, виконання яких потребує відносно багато часу, мають мати індикатор ходу виконання запиту;

— додавання записів у підлеглі таблиці повинно супроводжуватися помітно запису у головній таблиці з яким буде встановлений зв'язок, та запитом на підтвердження вибору запису у головній таблиці;

— при завантаженні програма повинна встановлювати необхідний графічний режим;

— програма повинна мати довідкову систему по використанню програми;

— надавати можливість робити закладки у таблицях та переходити до них завантаження програми;

— забезпечувати автоматичний пошук даних у зв'язаних таблицях.

Діаграми потоків даних (DFD) є одним з основних засобів моделювання функціональних вимог системи. З їхньою допомогою ці вимоги розбиваються на функціональні компоненти (процеси) і представляються у вигляді мережі, зв'язаної потоками даних. Головна мета таких засобів - продемонструвати, як кожен процес перетворить свої вхідні дані у вихідні, а також виявити відносини між цими процесами.

На діаграмах функціональні вимоги представляються за допомогою процесів і сховищ, зв'язаних потоками даних.

Потоки даних є механізмами, що використаються для моделювання передачі інформації з однієї частини системи в іншу. Потоки на діаграмах звичайно зображуються іменованими стрілками, орієнтація яких указує напрямок руху інформації.

Іноді інформація може рухатися в одному напрямку, оброблятися й повертатися назад у її джерело. Така ситуація може моделюватися або двома різними потоками, або одним, який спрямований в обидві сторони [2].

Призначення процесу складається в продукуванні вихідних потоків із вхідних відповідно до дії, що задає ім'ям процесу. Це ім'я повинне містити дієслово в невизначеній формі з наступним доповненням. Крім того, кожен процес повинен мати унікальний номер для посилань на нього усередині діаграми. Цей номер може використатися разом з номером діаграми для одержання унікального індексу процесу у всій моделі.

Сховище (накопичувач) даних дозволяє на певних ділянках визначати дані, які будуть зберігатися в пам'яті між процесами. Фактично сховище представляє "зрізи" потоків даних у часі. Інформація, що воно містить, може використатися в будь-який час після її визначення, при цьому дані можуть вибиратися в будь-якому порядку. Ім'я сховища повинне ідентифікувати його вміст і бути іменником. У випадку, коли потік даних входить або виходить в/зі сховища, і його структура відповідає структурі сховища, він повинен мати те ж саме ім'я, що немає необхідності відображати на діаграмі. Зовнішня сутність (або термінатор) представляє сутність поза контекстом системи, що є джерелом або приймачем системних даних.

Перед побудовою контекстної діаграми необхідно проаналізувати предметну область і виділити зовнішні сутності, що роблять вплив на розроблювальну систему.

У процесі роботи з розроблювальною системою беруть участь наступні зовнішні сутності: диспетчер відділу та бухгалтер відділу по розрахункам з

абонентами. Вони не тільки взаємодіють з системою, але також визначають її границі і зображуються на початковій контекстній діаграмі як зовнішні сутності. Початкова контекстна діаграма зображена на рисунку 1.6.

Рис. 1.6. Початкова контекстна діаграма

Для більш детального представлення роботи системи розглянемо наступний рівень діаграми. На рисунку 1.7 представлена діаграма першого рівня.

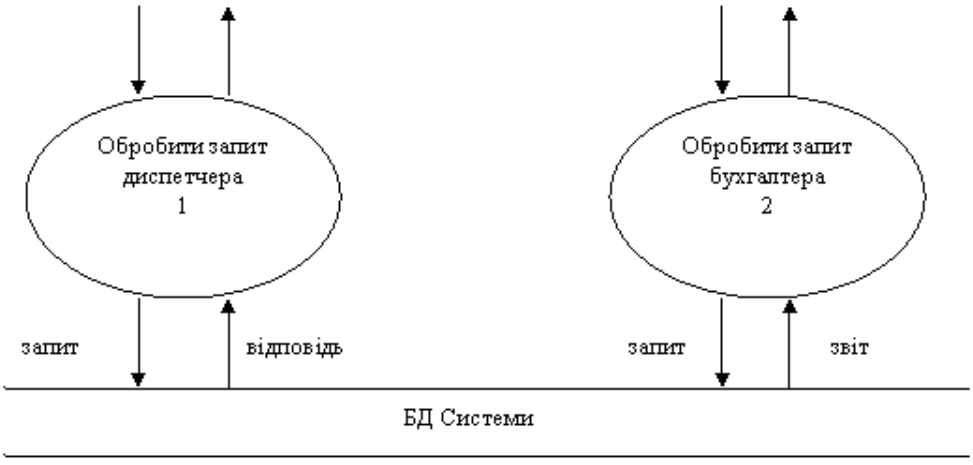

Рис. 1.7. Контекстна діаграма першого рівня

На рисунку 1.8 та рисунку 1.9 представлені контекстні діаграми другого рівня деталізацій процесів «Обробити запит диспетчера» та «Обробити запит бухгалтера» відповідно.

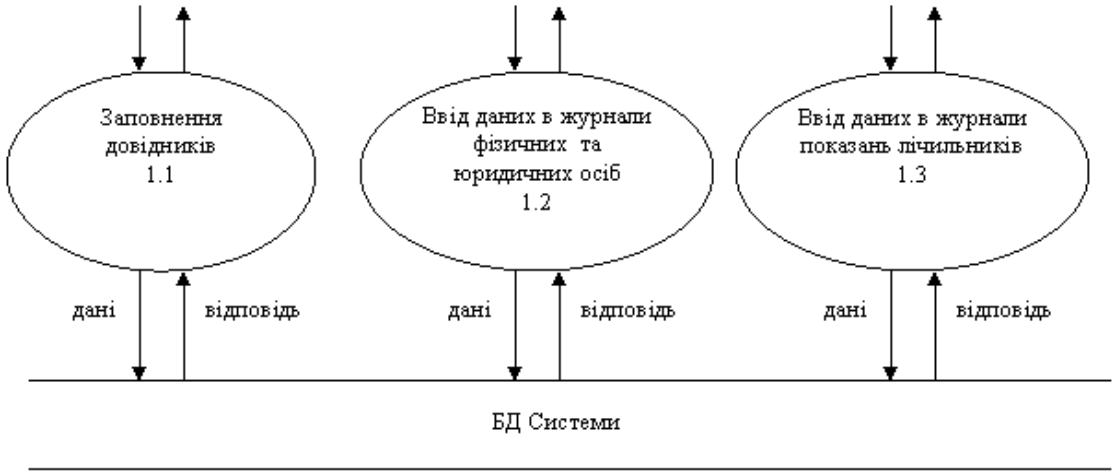

Рис. 1.8. Контекстна діаграма другого рівня «Обробити запит диспетчера»

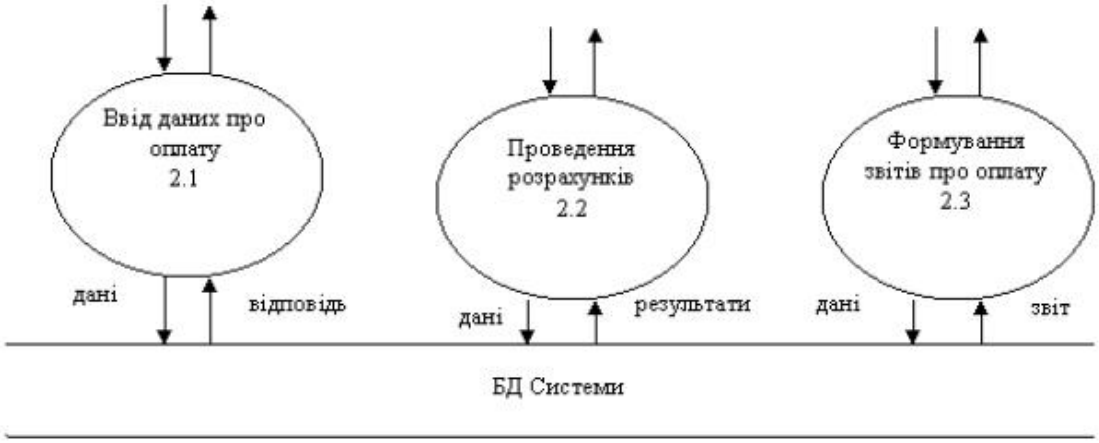

Рис. 1.9. Контекстна діаграми другого рівня «Обробити запит бухгалтера»

Для подальшої деталізації контекстних діаграм використовуємо діаграми більш низького рівня. На рисунку 1.10 представлена контекстна діаграма третього рівня, яка деталізує процес «Проведення розрахунків».

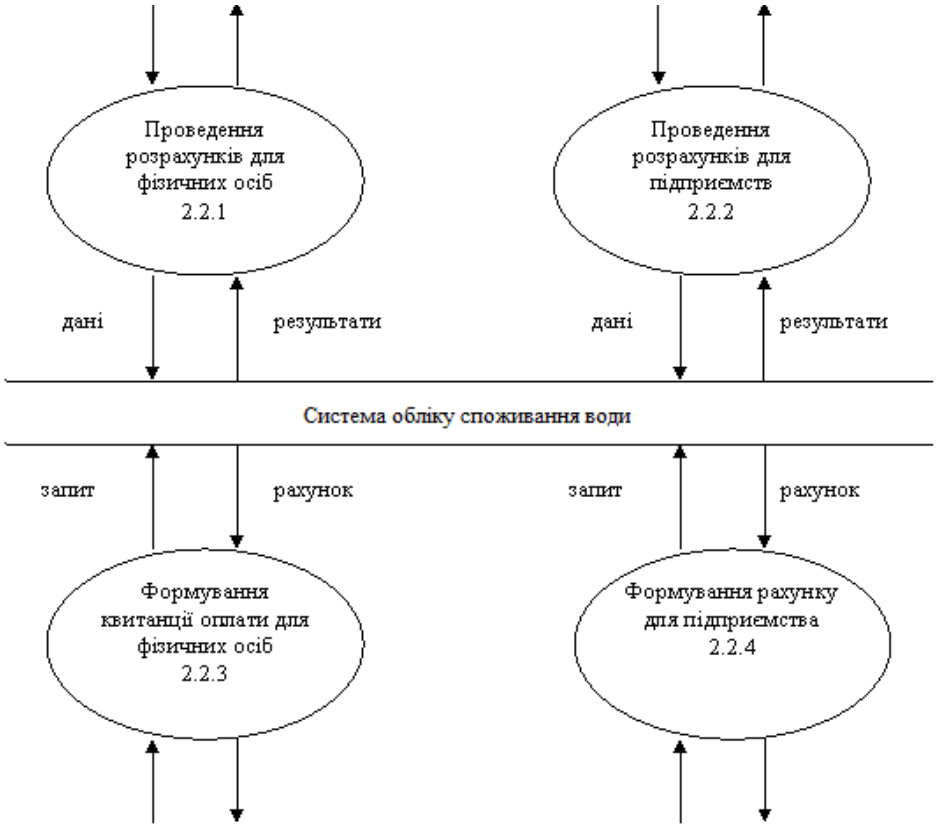

Рис. 1.10. Деталізація процесу «Проведення розрахунків»

На рисунку 1.11 представлена діаграма четвертого рівня, яка деталізую процес «Проведення розрахунків для фізичних осіб».

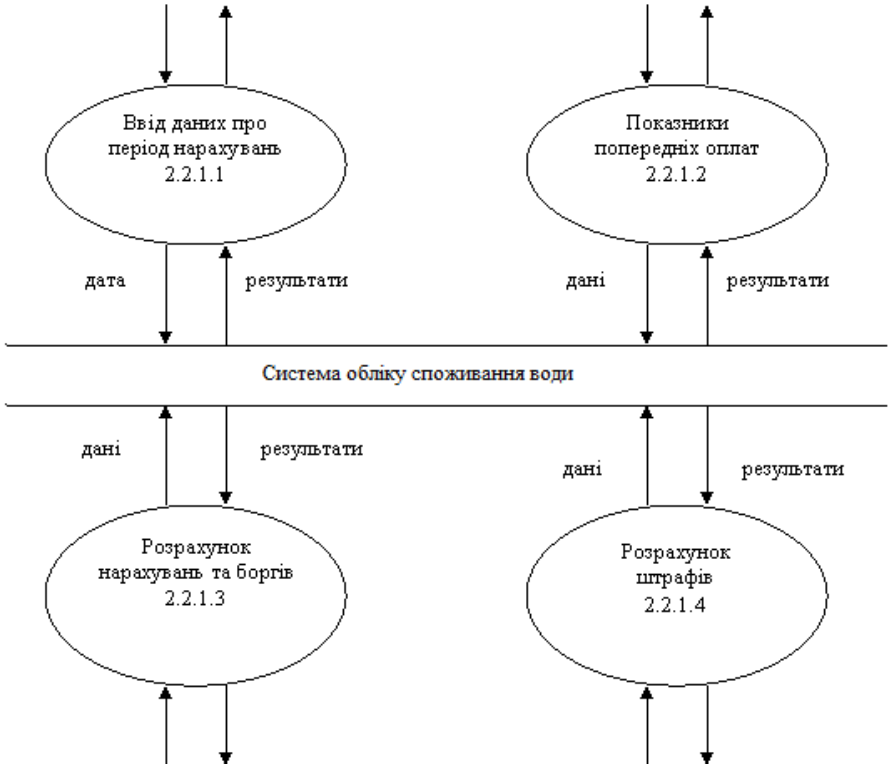

Рис. 1.11. Деталізація процесу «Проведення розрахунків для фізичних осіб»

На рисунку 1.12 представлена деталізація процесу 2.2.1.3 «Розрахунок нарахувань та боргів» у вигляді контекстної діаграми п’ятого рівня. Можливо продовження деталізації будь-якого процесу за допомогою діаграм більш низького рівня.

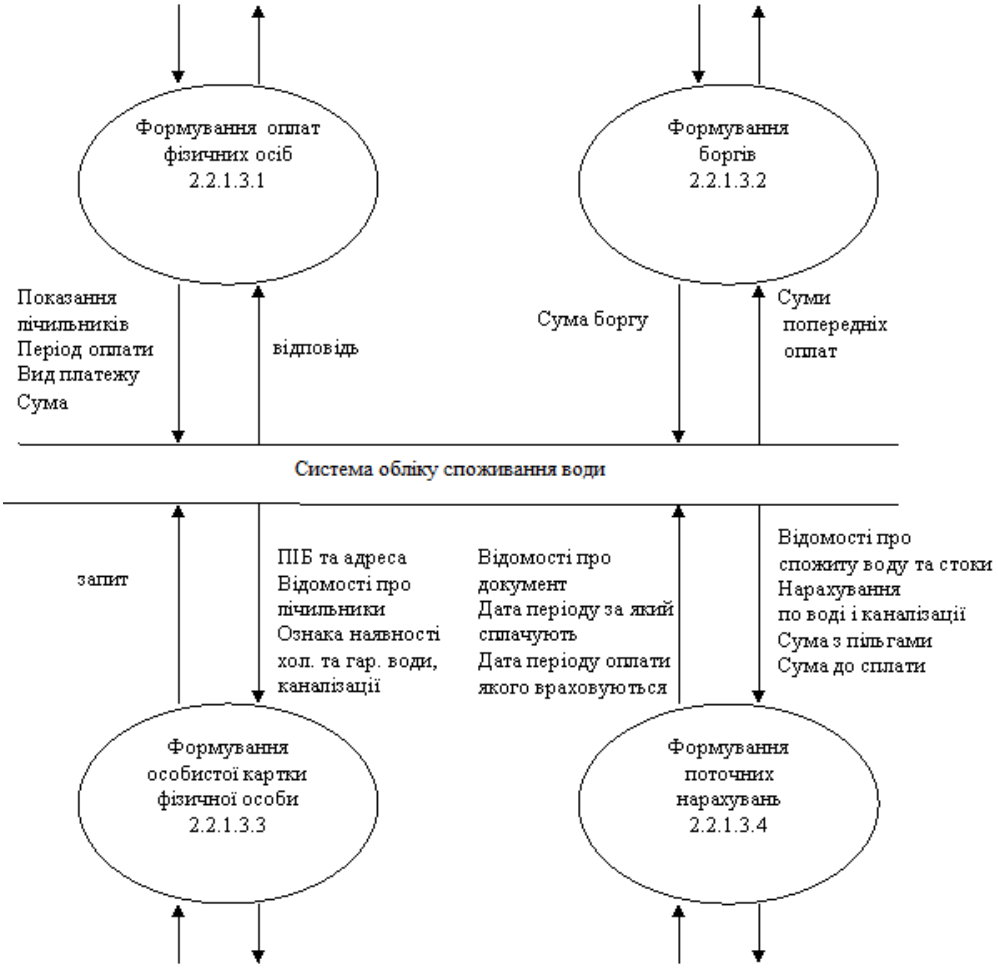

Рис. 1.12. Деталізація процесу «Розрахунок нарахувань та боргів»

Специфікація вимог до програмної системи – це специфікація окремого програмного продукту, програми або набору програм, які виконують деякі дії в деякому середовищі. Тобто – це повний опис поведінки системи що розробляється [3].

В загальному випадку специфікація включає наступне:

- глосарій проекту;
- опис варіантів використання.

Наведемо список основних термінів та понять в області розробки програмної системи «Управління персоналом» – глосарій. Глосарій – список понять в специфічній області знання з їх визначеннями [3]. Ці поняття та визначення подано у таблиці 1.2.

Таблиця 1.2

Глосарій

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту	
Клієнт	Особа, яка бере участь в економіко-правових відносинах з організацією
Договір про постачання води	Основний документ, що підтверджує відносини між клієнтом та організацією.
Облік води	Показники, які характеризують споживання води окремим клієнтом на конкретну дату чи час.
Програмне забезпечення	сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації цих програм [3].
Програмний продукт	програмний засіб, програмне забезпечення, які призначені для постачання користувачів (покупцеві, замовникові) [3].
Бізнес-процес	будь-яка діяльність, що має вхідний продукт, додає вартість до нього, та забезпечує вихідний продукт для внутрішнього або зовнішнього споживача [3].
База даних	логічно впорядкований та взаємопов’язаний набір даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.
2. Користувачі системи	
Адміністратор	Особа, яка займається безпосереднім управлінням системою на технічному рівні.
Адміністратор БД	Особа, яка займається безпосереднім управлінням БД.
Начальник відділу	Особа, яка здійснює управлінські функції відділу обліку споживання води
Спеціаліст відділу	Особа, яка здійснює облікову роботу на підприємстві, яка стосується оформлення показників, ведення документації, та формування заборгованості

3. Вхідні та вихідні документи	
База даних	логічно впорядкований та взаємопов’язаний набір даних, що використовуються спільно та призначені для задоволення інформаційних потреб користувачів.
Договір про надання послуг	юридичний документ, в якому визначено початок надання послуг клієнту
Договір про припинення надання послуг	юридичний документ, в якому визначено завершення надання послуг клієнту
Список боржників	документ, який забезпечує реєстрацію надходження квитанцій про оплату, на основі яких можна сформувати список боржників

У таблицях 1.3 – 1.13 наведено опис варіантів використання, що реалізують основну функціональність програмної системи, а на рисунку 1.13 представлено діаграму варіантів використання системи.

Таблиця 1.3

Варіант використання «Формуння журналів фізичних та юридичних осіб»

Контекст використання	Управління клієнтами.
Дійові особи	Диспетчер
Передумова	Користувач аутентифікований та авторизований.
Тригер	Натиснення кнопки «Управління журналами».
Сценарій	1. Заповнення полів. 2. Перевірка правильності введених даних. 3. Натиснення кнопки «Зберегти».
Постумова	Відображення інформації з відповідного журналу.

Таблиця 1.4

Варіант використання «Формування журналів показань лічильника»

Контекст використання	Управління клієнтами.
Дійові особи	Диспетчер
Передумова	Користувач аутентифікований та авторизований.
Тригер	Натиснення кнопки «Внести показники».
Сценарій	1. Заповнення полів. 2. Перевірка правильності введених даних. 3. Натиснення кнопки «Зберегти».
Постумова	Відображення показників лічильника для клієнта.

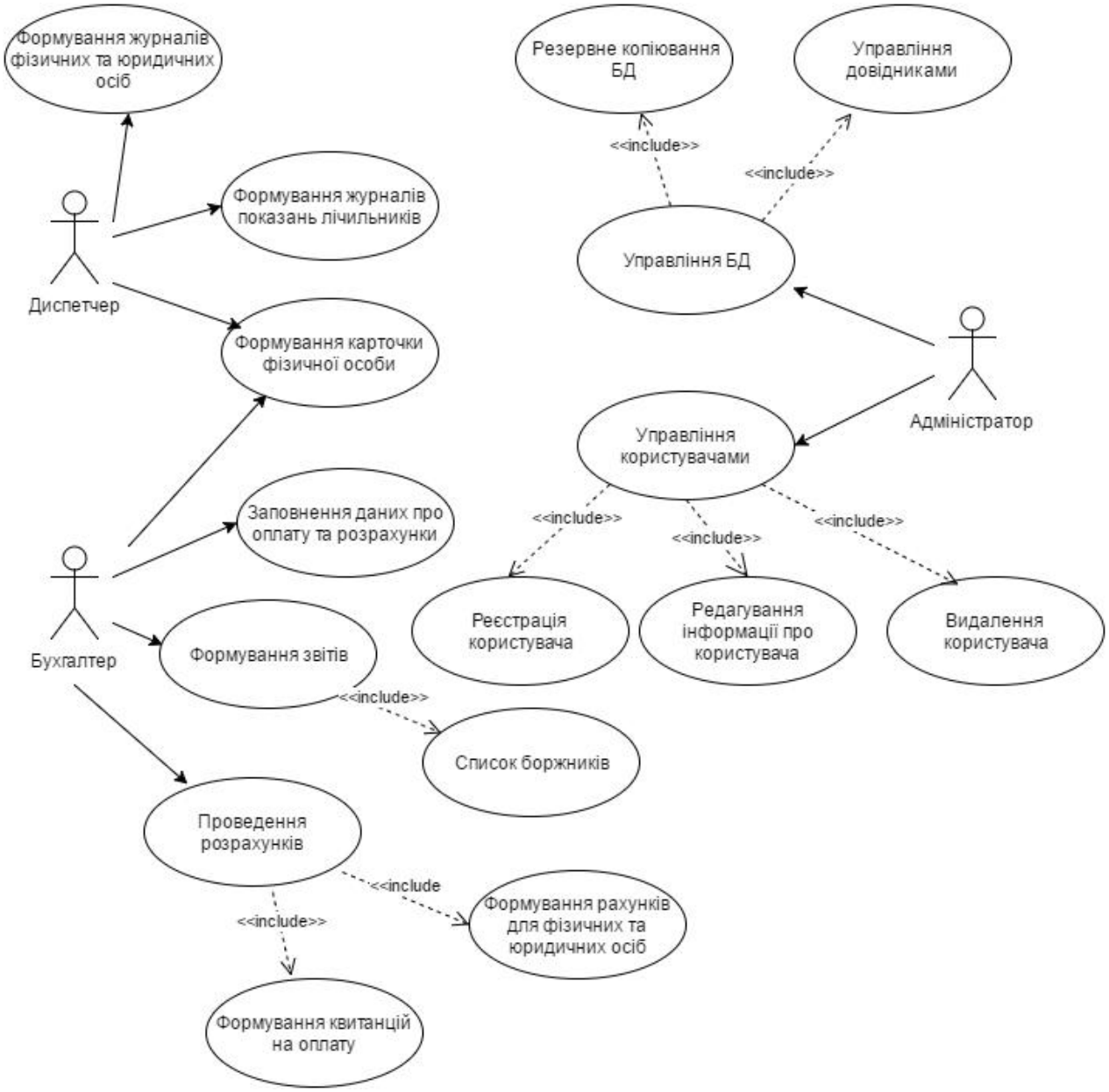

Рис. 1.13. Діаграма варіантів використання системи обліку споживання води

Таблиця 1.5

Варіант використання «Формування карточки фізичної особи»

Контекст використання	Управління клієнтами.
Дійові особи	Всі користувачі відповідно до прав доступу
Передумова	Користувач аутентифікований та авторизований.
Тригер	Натиснення кнопки «Створити картку».
Сценарій	1. Заповнення полів. 2. Перевірка правильності введених даних. 3. Натиснення кнопки «Зберегти».
Постумова	Відображення списку фізичних осіб

Таблиця 1.6

Варіант використання «Заповнення даних про оплату та розрахунки»

Контекст використання	Управління інформацією.
Дійові особи	Бухгалтер
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Оплата».
Сценарій	1. Заповнення полів. 2. Перевірка правильності введених даних. 3. Натиснення кнопки «Зберегти».
Постумова	Відображення даних.

Таблиця 1.7

Варіант використання «Проведення розрахунків»

Контекст використання	Управління розрахунками
Дійові особи	Бухгалтер
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Розрахунки».
Сценарій	1. Вибір розрахунку. 2. Заповнення необхідних полів. 3. Перевірка введених даних. 4. Натиснення кнопки «Зберегти».
Постумова	Відображення даних.

Таблиця 1.8

Варіант використання «Формування звітів»

Контекст використання	Управління звітністю
Дійові особи	Бухгалтер
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Звітність».
Сценарій	1. Вибір звіту. 2. Заповнення необхідних полів. 3. Перевірка введених даних. 4. Натиснення кнопки «Софрмувати».
Постумова	Відображення звіту.

Таблиця 1.9

Варіант використання «Список боржників»

Контекст використання	Управління клієнтами.
Дійові особи	Начальник відділу
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Облік споживання води».
Сценарій	1. Вибір пунктів звіту. 2. Формування звіту. 3. Перевірка введених даних. 4. Натиснення кнопки «Зберегти».
Постумова	Відображення звіту за певним фільтром.

Таблиця 1.10

Варіант використання «Управління БД»

Контекст використання	Управління БД.
Дійові особи	Адміністратор
Передумова	Користувач аутентифікований та авторизований.
Тригер	Перехід в системний розділ
Сценарій	1. Вибір дати. 2. Натиснення кнопки «Резервне копіювання». 3. Підтвердження операції.
Постумова	Відображення списку копій БД.

Таблиця 1.11

Варіант використання «Реєстрація користувача»

Контекст використання	Управління користувачами.
Дійові особи	Адміністратор.
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Користувачі».
Сценарій	1. Вибір фотокарточки користувача. 2. Заповнення необхідних полів. 3. Перевірка введених даних. 4. Натиснення кнопки «Зберегти».
Постумова	Відображення списку користувачів.

Таблиця 1.12

Варіант використання «Редагування інформації про користувача»

Контекст використання	Управління користувачами.
Дійові особи	Адміністратор.
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Користувачі».
Сценарій	1. Вибір користувача. 2. Натиснення кнопки «Редагувати». 3. Заповнення необхідних полів. 4. Перевірка введених даних. 5. Натиснення кнопки «Зберегти».
Постумова	Відображення списку користувачів.

Таблиця 1.13

Варіант використання «Видалення користувача»

Контекст використання	Управління користувачами.
Дійові особи	Адміністратор.
Передумова	Користувач аутентифікований та авторизований.
Тригер	Відкриття розділу «Користувачі».
Сценарій	1. Вибір користувача. 2. Натиснення кнопки «Видалити». 3. Підтвердження операції видалення.
Постумова	Відображення списку користувачів.

В таблиці 1.14 наведено специфікацію функціональних вимог програмної системи.

Таблиця 1.14

Специфікація функціональних вимог

Іденти-фікатор вимоги	Назва вимоги	Атрибути вимоги		
		Пріоритет	Складність	Контакт
1	Формуння журналів фізичних та юридичних осіб	обов’язкове	висока	Диспетчер
2	Формування журналів показань лічильника	обов’язкове	висока	Диспетчер
3	Формування карточки фізичної особи	обов’язкове	висока	Диспетчер Бухгалтер
4	Заповнення даних про оплату та розрахунки	рекомендоване	середня	Бухгалтер
5	Проведення розрахунків	обов’язкове	висока	Бухгалтер
6	Формування звітів	обов’язкове	висока	Бухгалтер
7	Список боржників	обов’язкове	висока	Бухгалтер
8	Управління БД	опційне	висока	Адміністратор
9	Реєстрація користувача	обов’язкове	висока	Адміністратор
10	Редагування інформації про користувача	обов’язкове	висока	Адміністратор
11	Видалення користувача	обов’язкове	висока	Адміністратор

Специфікацію нефункціональних вимог наведено в таблиці 1.15.

Наведемо специфікацію суттєвих для проекту нефункціональних вимог:

1. Застосовність:
   - мінімальний час для навчання звичайних і досвідчених користувачів;
   - відповідність стандартам графічного інтерфейсу.
2. Надійність:
   - постійна безвідмовна робота;
   - пропускна здатність каналу зв’язку 100 Mb/s;
   - забезпечення можливості віддаленого доступу до комп’ютера, на якому буде встановлена система;
   - доступність – 5%.
3. Робочі характеристики:
   - швидкість завантаження інтернет-ресурсу: 0,1 – 1 с;

- число транзакцій: 100 / 1 с;
- використання ресурсів: від 1 Gb, в залежності від кількості клієнтів.

4. Проектні обмеження:
   - Операційна система Microsoft Windows 7/8;
   - MySQL;
   - Eclipse IDE for Java EE Developers;
5. Вимоги до документації
   - наявність інтерактивної довідки.
6. Інтерфейси:

   -інтерфейс користувача – Java-інтерфейс.

Таблиця 1.15

Специфікація нефункціональних вимог

Ідентифікатор вимоги	Назва вимоги	Атрибути вимог		
		Пріоритет	Складність	Контакт
Застосовність				
1.1	Час, необхідний для навчання звичайних і досвідчених користувачів	Рекомендована	Низька	Адміністратор
1.2	Основні вимоги застосовності нової системи відносно інших систем, які знають користувачі	Опційна	Низька	Адміністратор
1.3	Вимоги по відповідальності стандартам графічного інтерфейсу користувача	Рекомендована	Низька	Адміністратор
Надійність				
2.1	Доступність	Обов’язкова	Середня	Адміністратор
2.2	Середній час безвідмовної роботи	Рекомендована	Середня	Адміністратор
2.3	Точність	Обов’язкова	Середня	Адміністратор
Робочі характеристики				
3.1	Використання ресурсів	Рекомендована	Середня	Адміністратор
4.1	Вимоги до технології програмування	Рекомендована	Середня	Адміністратор

Висновки до розділу 1

Здійснено опис предметної області, напрями діяльності. Визначено склад функцій, що входять до бізнес-процесу на основі яких розроблено схему управління бізнес-процесом. Проведено аналіз відомих програмних систем обліку споживання води. Здійснено аналіз вимог до програмної системи.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ ОБЛІКУ СПОЖИВАННЯ ВОДИ ТЕПЛОМЕРЕЖЕЮ

2.1. Розроблення архітектури програмної системи

Для вирішення даної бізнес проблеми, була вибрана клієнт-серверна архітектура. Вона найбільш придатна для задач саме такого типу, коли потрібно забезпечити доступ до системи багатьом користувачам. Найголовнішим є те, що клієнт-серверна архітектура надає можливість віддаленого доступу [4].

Проект, який побудований на клієнт-серверній архітектурі, повинен складатися з трьох частин: зв'язок з базою даних, відображення даних клієнту та бізнес логіка проекту яка обробляє запити користувача і відображає саме ту інформацію, яку захотів користувач. Обробка та збереження даних відбувається на боці сервера, відображення даних і надсилання запитів на їхню модифікацію виконується на боці клієнта.

Статичний аспект такої архітектури зображений, за допомогою наступної діаграми, на рисунку 2.1.

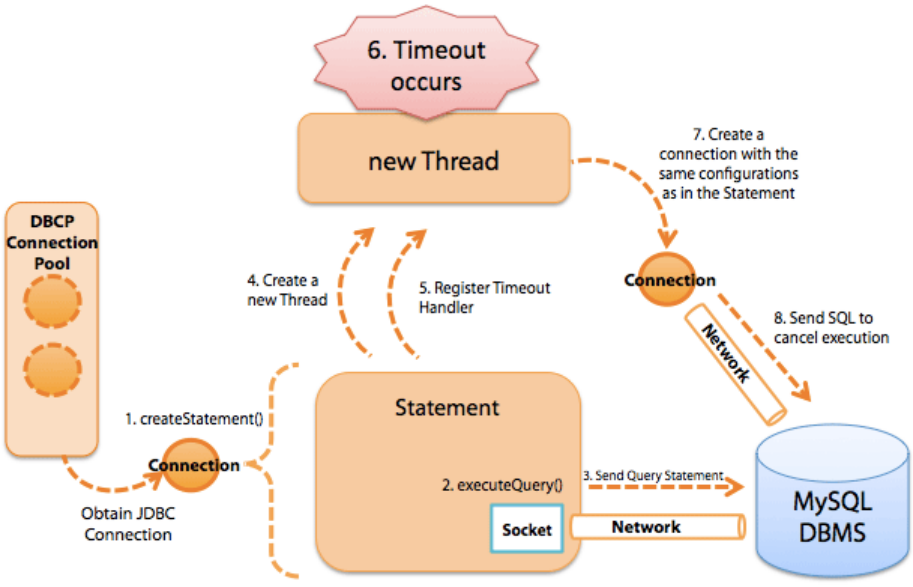

Рис. 2.1. Реалізації клієнт-серверної архітектури

На цій діаграмі клієнтською аплікацією виступає клієнт, розроблений на основі, тобто розширення технології Java. Він, у свою чергу, віддає на сервер запити, відповідно до того з якими формами працює працівник відділу обліку споживання води. Реалізація взаємодії із базою даних здійснюється за допомогою JDBC connection.

Уся бізнес-логіка проекту виконується на другому ріні. Вся обробка запитів від клієнта і надсилання йому відповіді здійснюється на цьому рівні. Також тут визначається, які запити повинні іти до бази даних. Третій рівень – це власне сама база даних, яка приймає Sql-запити, і у відповідності їм повертає дані.

На рисунку 2.2 представлено узагальнена архітектура програмної системи, яка реалізована як клієнт-серверна технологія.

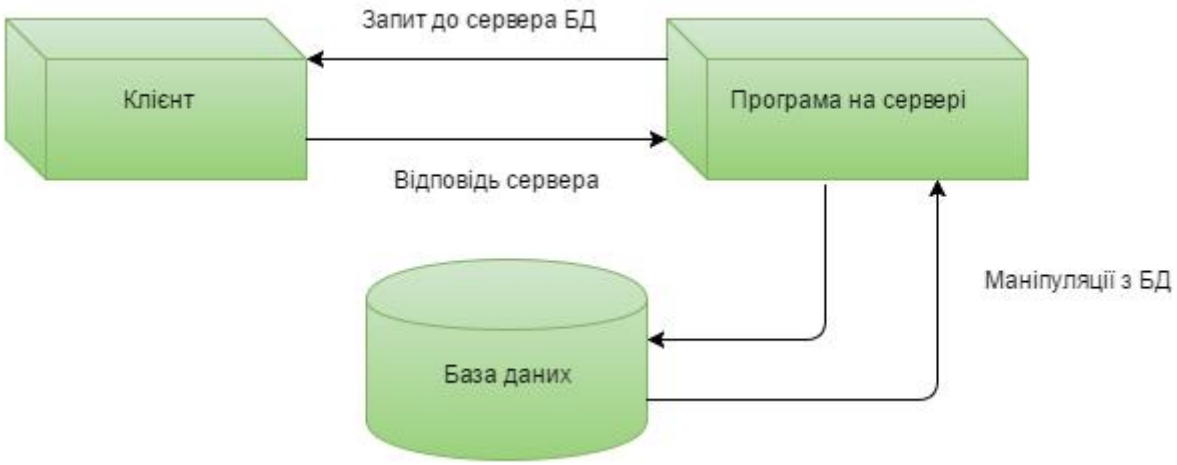

Рис. 2.2. Архітектура системи

Наявність комп’ютерів у кожній структурній одиниці дозволяє накопичувати значну кількість інформації в електронному вигляді та організовувати діяльність на належному рівні.

У зв’язку зі значним об’ємом інформації, яку використовує організація, а саме «Тернопільводоканал», а також зростанням потреби пошуку конкретних, необхідних в певний момент чи в певній ситуації, даних, виникає необхідність якісної комп’ютеризації та автоматизації діяльності установи, впровадження новітніх інформаційних систем. Адже інформаційні технології здатні

поєднувати у собі технічні можливості обчислювальної техніки, засобів зв’язку, інформатики, вони спрощують роботу зі збору, накопичення, оброблення, аналізу, доставки інформації споживачам. За допомогою інформаційних систем та технологій значно легше здійснити автоматизацію рутинних операцій управління, підготовку аналітичної інформації для прийняття управлінських рішень, а також забезпечити нові види інформаційного обслуговування.

Структуру розроблювальної системи можна представити у виді організаційної і функціональної структури. В організаційній структурі системи можна виділити найбільш важливі наступні складові:

Розроблювальна програма – клієнт - надає доступ до бази даних, забезпечує обробку даних, формує різні способи відображення даних;

Робоча станція – комп’ютер призначений для встановлення на нього програми-клієнта та її функціонування;

Мережа передачі даних - побудована на основі сучасних телекомунікаційних технологій, зв'язує програми-клієнти з сервером;

Програма-сервер - забезпечує функціонування архітектури клієнт - сервер, забезпечуючи доступ до бази даних;

Комп'ютер-сервер - забезпечує розташування програми-сервера та бази даних.

З функціональної точки зору система підрозділяється на три ієрархічних рівні:

- рівень збереження даних - база даних розташована на сервері, у якій буде зберігатися вся необхідна інформація;
- рівень передачі даних буде представлений мережею передачі даних;
- рівень обробки і подання даних - буде існувати у виді програм-клієнтів які будуть у процесі роботи надавати доступ до бази даних, здійснювати обробку і відображення даних за допомогою графічного інтерфейсу користувача.

Узагальнена блок-схема алгоритму роботи програми представлена на рисунку 2.3.

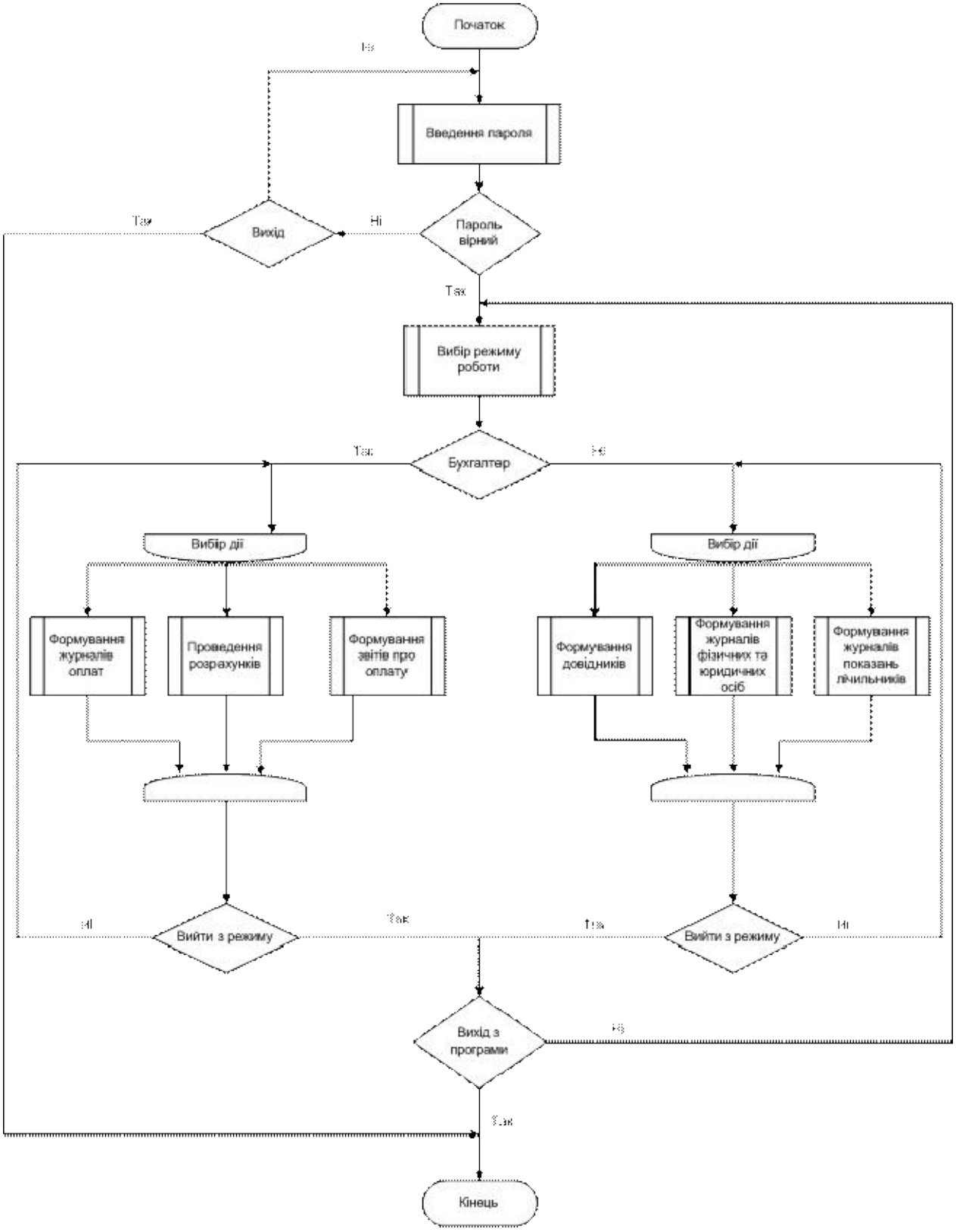

Рис. 2.3. Узагальнений алгоритм роботи системи

На рисунку 2.4 наведена UML діаграма станів. Слід зауважити, що в розроблювана модель діаграма станів є єдиною і описує поведінку системи. Головна перевага даної діаграми станів - можливість моделювати умовний характер реалізації всіх варіантів використання у формі зміни окремих станів розроблюваної системи.

Рис. 2.4. Діаграма станів

Діаграма, яка буде розглянута наступною - це діаграма активності, яка

конкретизує та деталізує всі дії та операції, які виконуються системою, а також відображає послідовність переходів від виконання однієї дії чи діяльності до виконання наступної. Дана діаграма для бухгалтера показана на рисунку 2.5.

Діаграм активності показує діяльність роботи працівника, яка функціонує наступним шляхом: спочатку клієнт звертається до системи; введення паролю; перевірка введеного паролю; якщо пароль введено вірно виконуються наступні дії: зміна даних; обробка даних в базі даних; відображення результату; якщо пароль не вірний:виведення повідомлення про помилку.

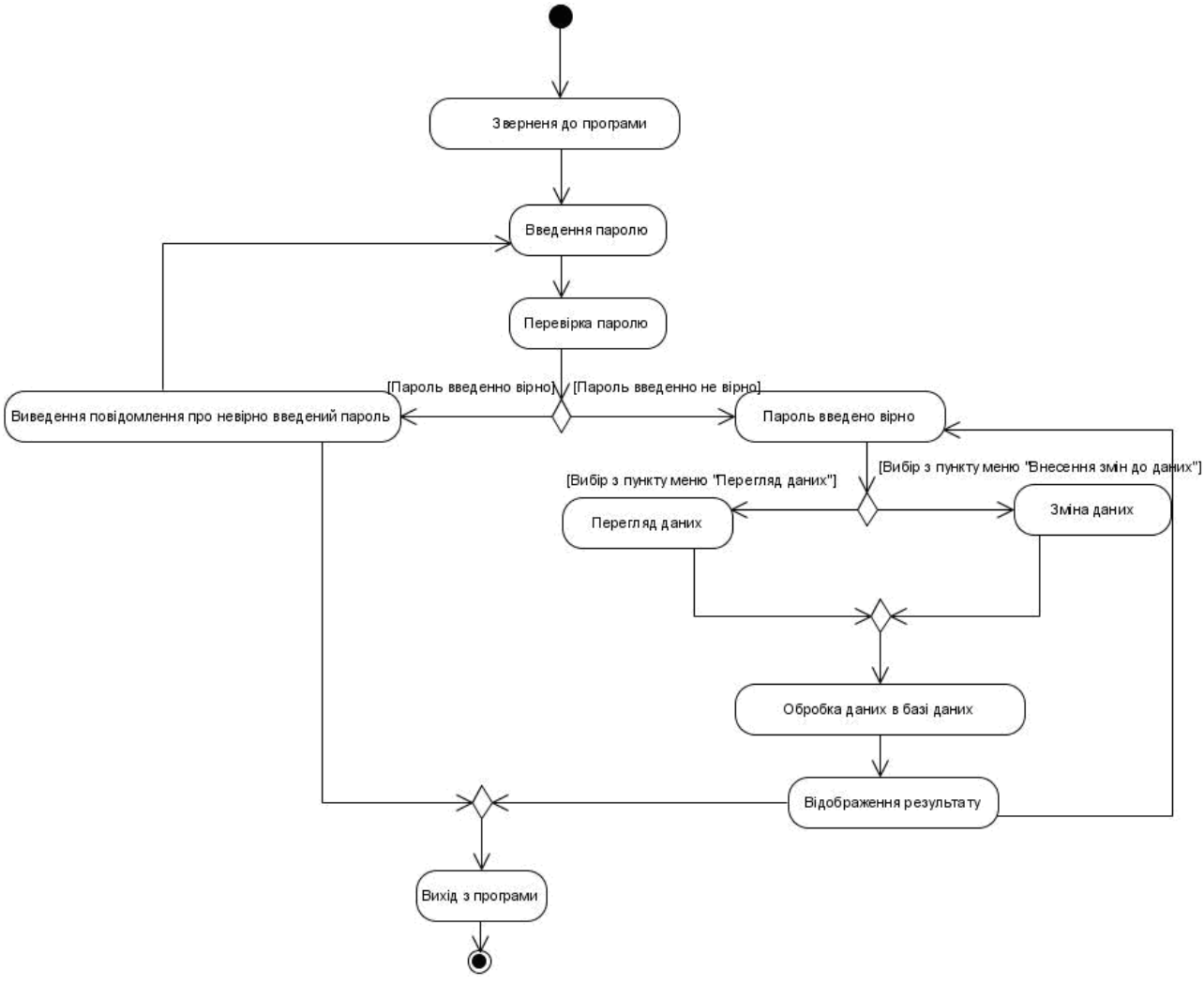

Рис. 2.5. Діаграма активності при вході в систему працівника

На рисунку 2.6 представлено діаграму розміщення системи із врахування особливостей використання.

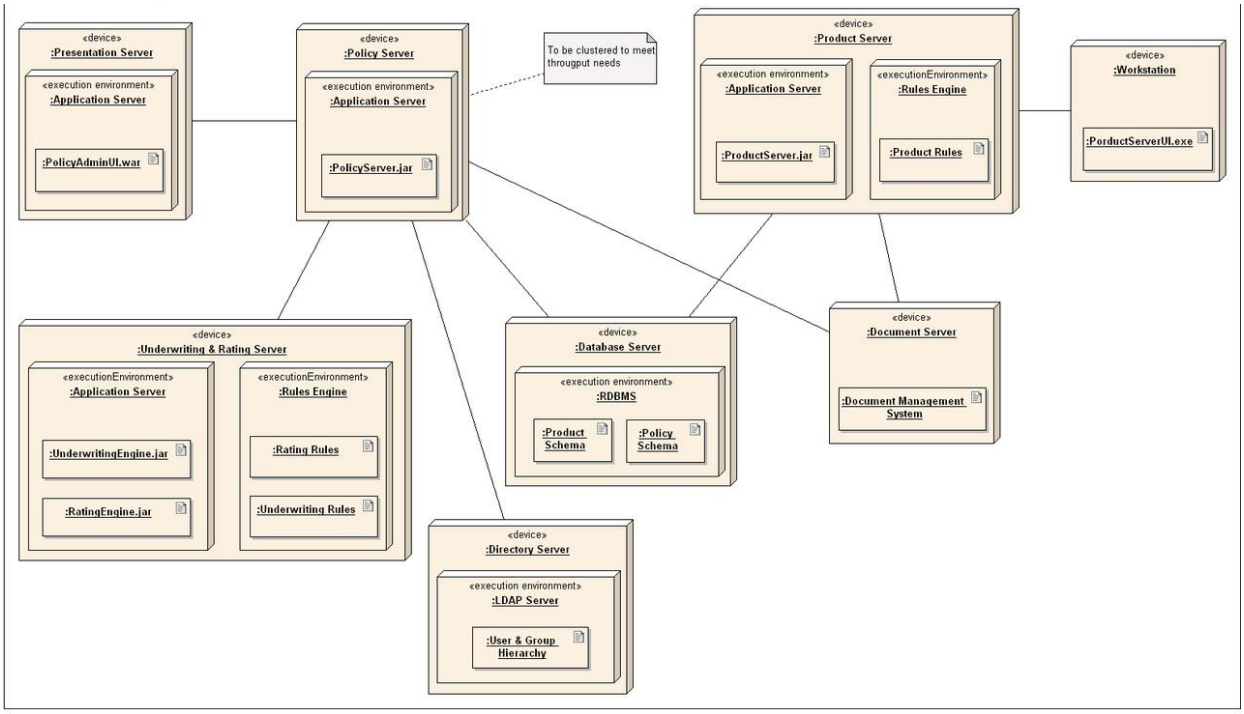

Рис. 2.6. Діаграма розміщення

На рисунку 2.7 представлено діаграму послідовності дій при вході в систему «Система обліку споживання води КП «Тернопільводоканал»».

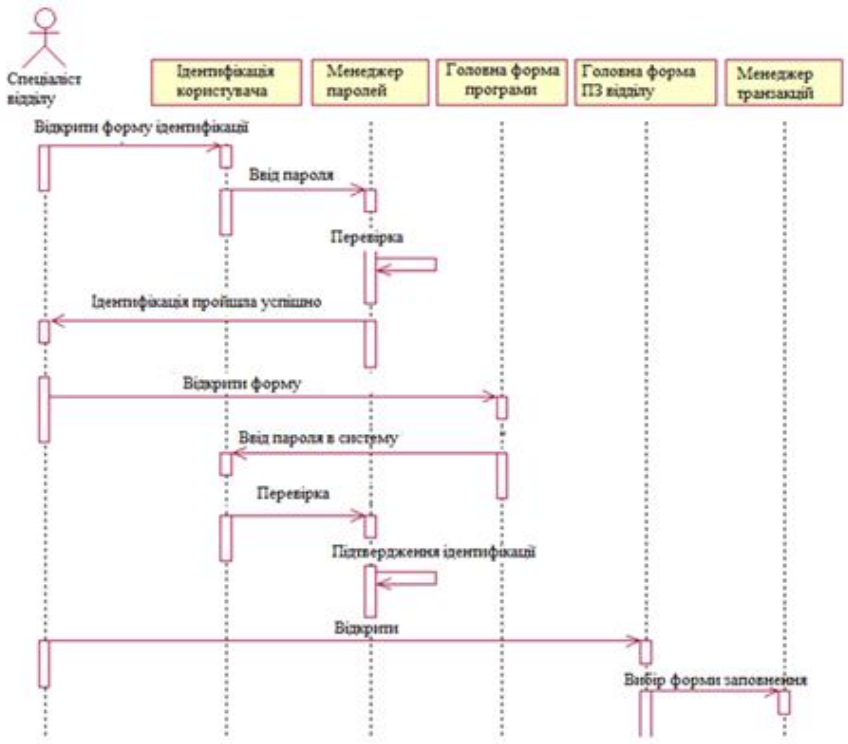

Рис. 2.7. Діаграма послідовності дій при вході в систему

Діаграма класів дозволяє створювати логічне представлення системи, на основі якого створюється вихідний код описаних класів. На діаграмах класів відображаються деякі класи і пакети системи. Це статичні картини фрагментів системи та зв'язків між ними [8].

Об'єднувати класи можна як завгодно, проте існує кілька найпоширеніших підходів. По-перше, можна групувати класи по стереотипу. У такому разі виходить один пакет з класами-сутностями, один з прикордонними класами, один з керуючими класами і так далі. Цей підхід може бути корисний з точки зору розміщення готової системи, оскільки всі знаходяться на клієнтських машинах прикордонні класи вже опиняються в одному пакеті.

Другий підхід полягає в об'єднанні класів по їх функціональності. Наприклад, у пакеті Security (Безпека) будуть міститися всі класи, що відповідають за безпеку програми. Перевага цього методу полягає в можливості повторного використання пакетів. Якщо уважно підійти до групування класів, можна отримати практично не залежать один від одного пакети. Наприклад, пакет Security можна використовувати і в інших програмах.

Нарешті, застосовують комбінацію двох зазначених підходів. Для подальшої організації класів дозволяється вкладати пакети один в одного. На високому рівні абстракції можна згрупувати класи по функціональності, створивши пакет Security. Всередині нього можна створити інші пакети, згрупувавши відповідальні за безпеку класи по функціональності або по стереотипу.

При створенні діаграм класів був використаний другий підхід до об’єднання. Всі діаграми класів об’єднані в пакети, що пов'язані з певним підрозділом або до функціональної належності.

Кожна діаграма класів тісно пов’язана з реалізаціями прецедентів, розглянутими вище. На рисунку 2.8 зображена діаграма класів для підсистеми доступу до бази даних, а загальна діаграма класів представлена на рисунку 2.10.

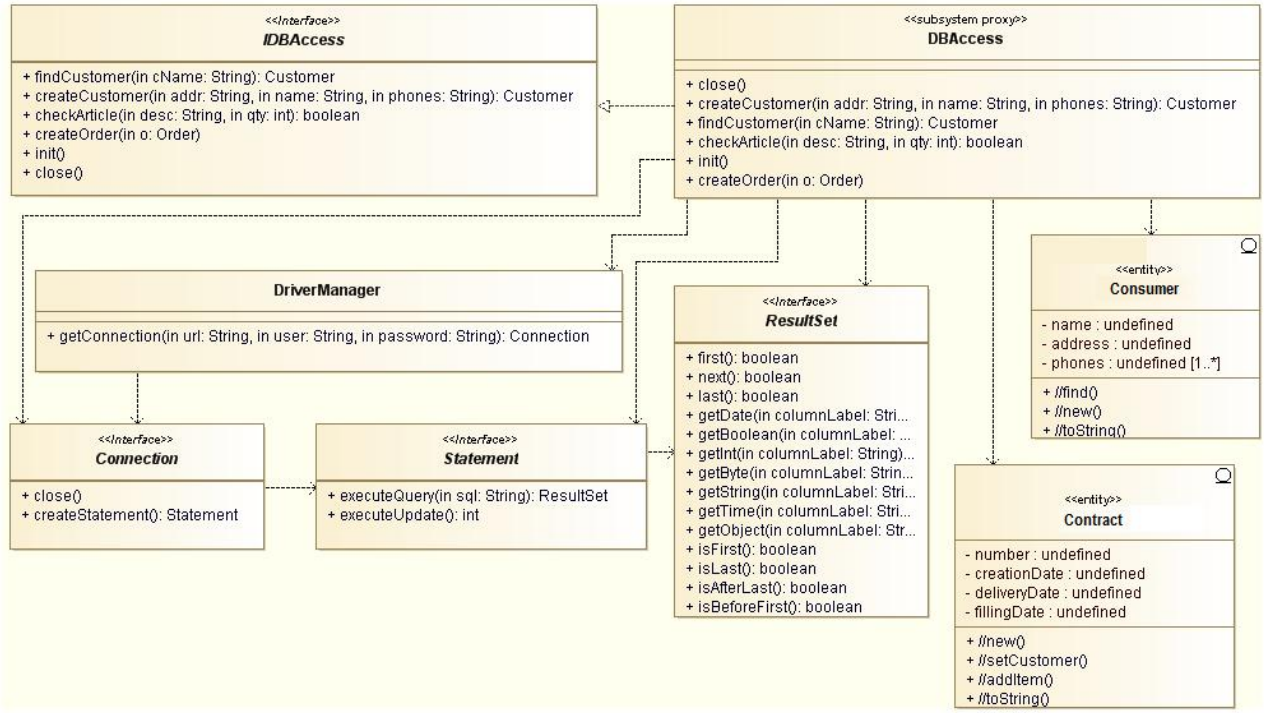

Рис. 2.8. Діаграма класів підсистеми DBAccess

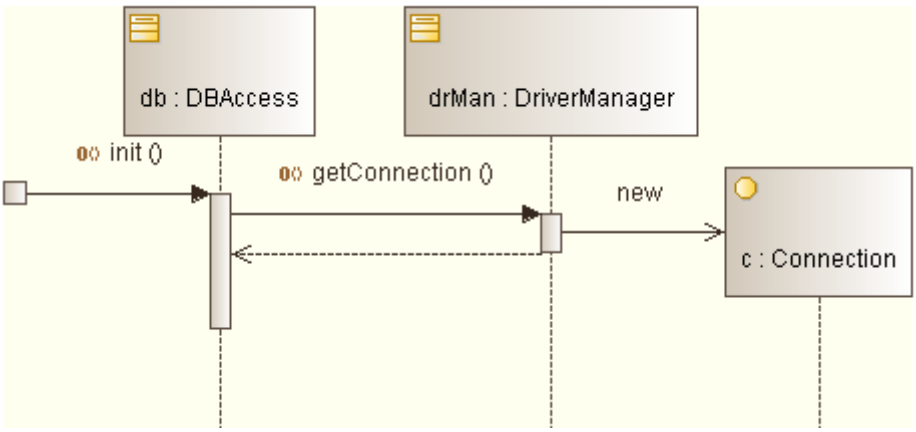

Рис. 2.9. Діаграма послідовності, що описує реалізацію операції init()

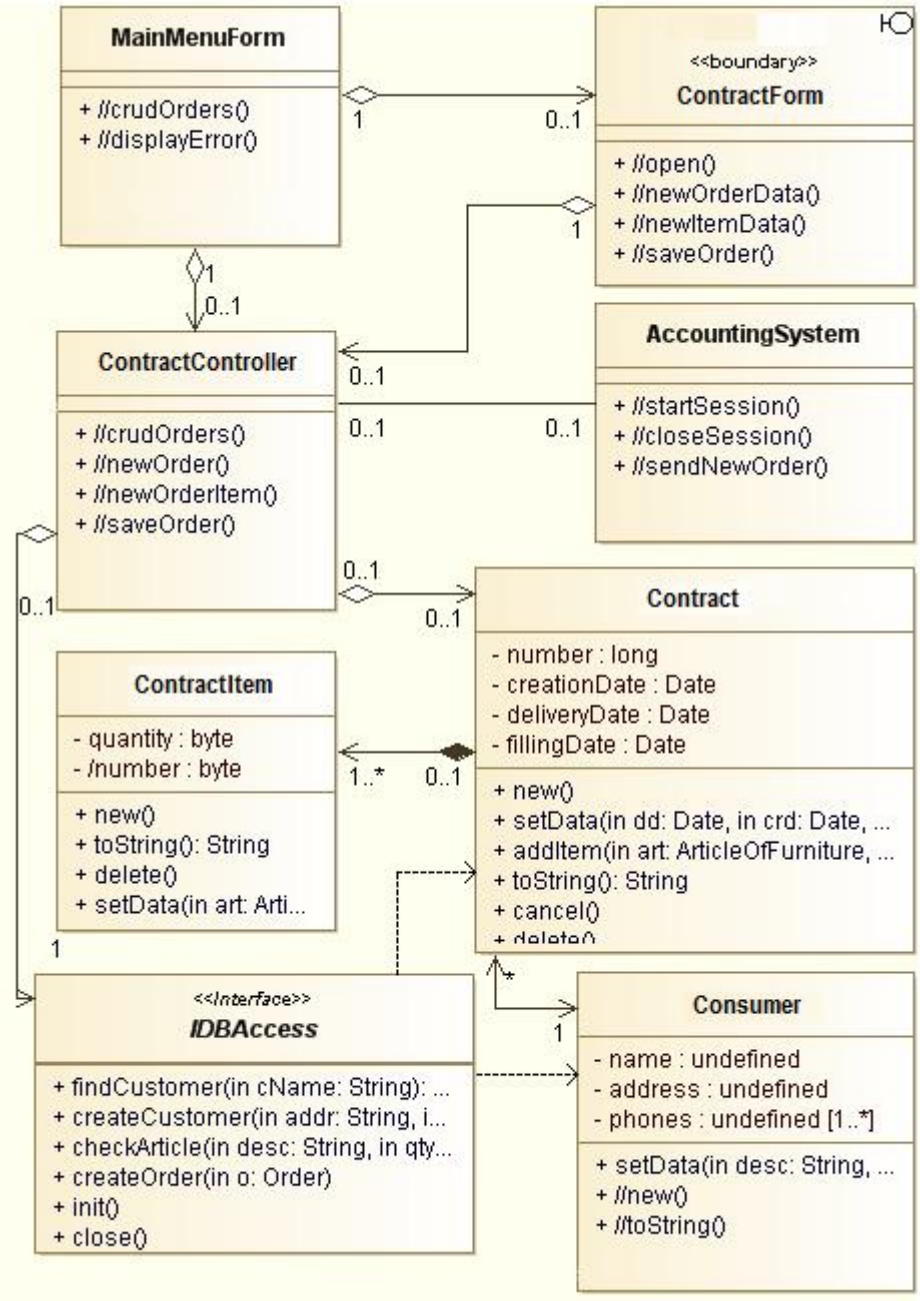

Рис. 2.10. Діаграма класів системи обліку споживання води

Кожен клас створювався на основі діаграм послідовностей, так як саме ці діаграми допомагають зрозуміти які класи необхідно створювати для ефективної роботи системи, а які краще опустити за непотрібністю.

Діаграми станів призначена для відображення станів об’єктів системи, які мають складну модель поведінки. Станом називається одне з можливих умов, в яких може існувати об’єкт [8]. Перебуваючи в конкретному стані, об’єкт може виконувати певні дії. Наприклад, може генерувати звіт, здійснювати деякі

обчислення або посилати подія іншому об'єкту. Зі станом можна пов'язувати дії п'яти типів: діяльність, вхідна дія, вихідна дія, подія і історія стану.

Кінцевий варіант діаграми станів представлений на рисунку 2.11 для класу «LoginForm», 2.12 для класу «Main», 2.13 для класу «ProgressForm» і 2.14 для класу «DBMS».

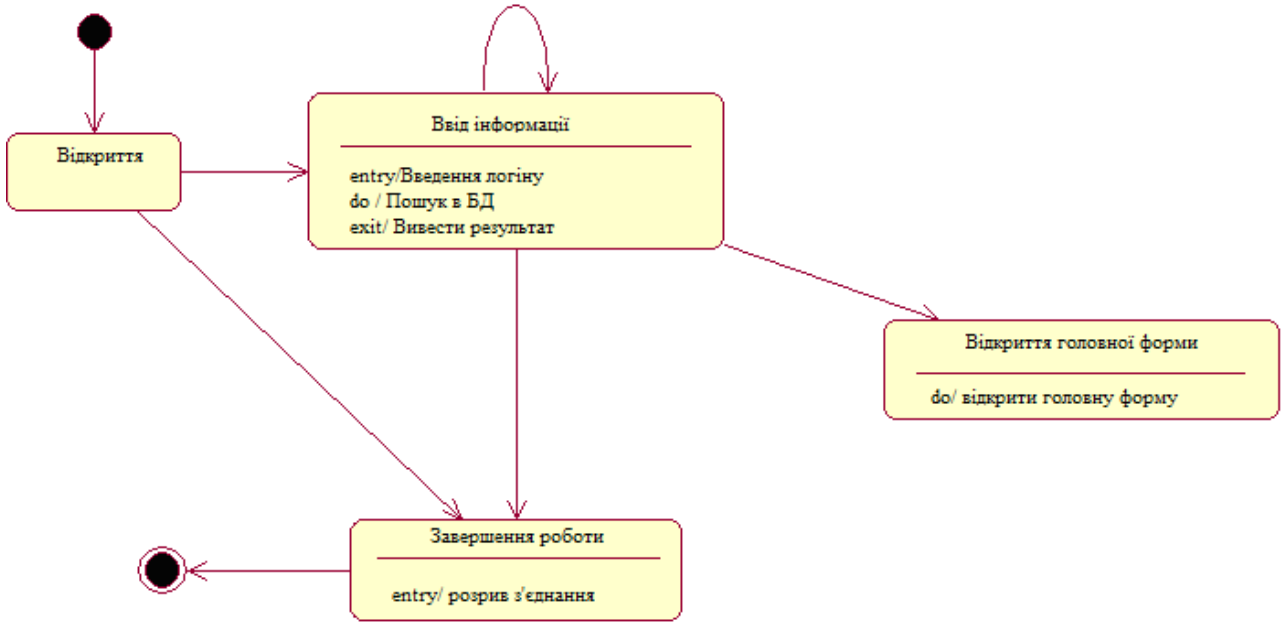

Рис. 2.11. Діаграма станів для класу «Login»

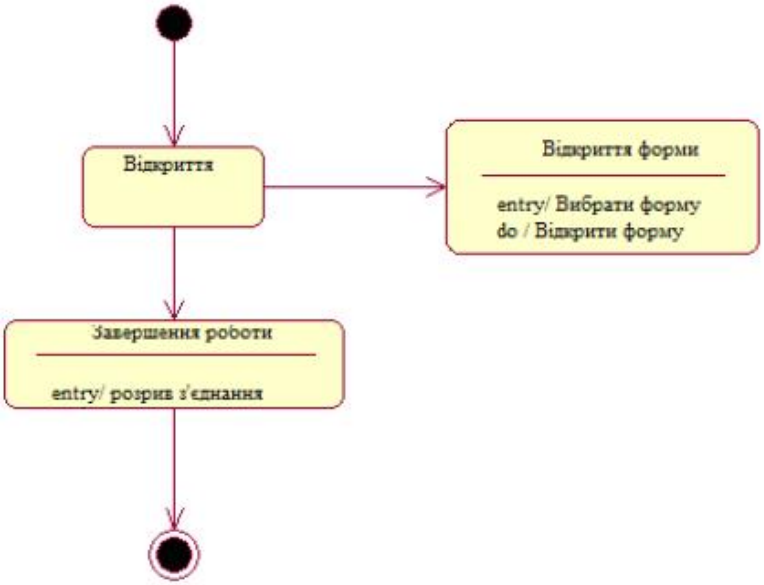

Рис. 2.12. Діаграма станів для класу «MainWindow»

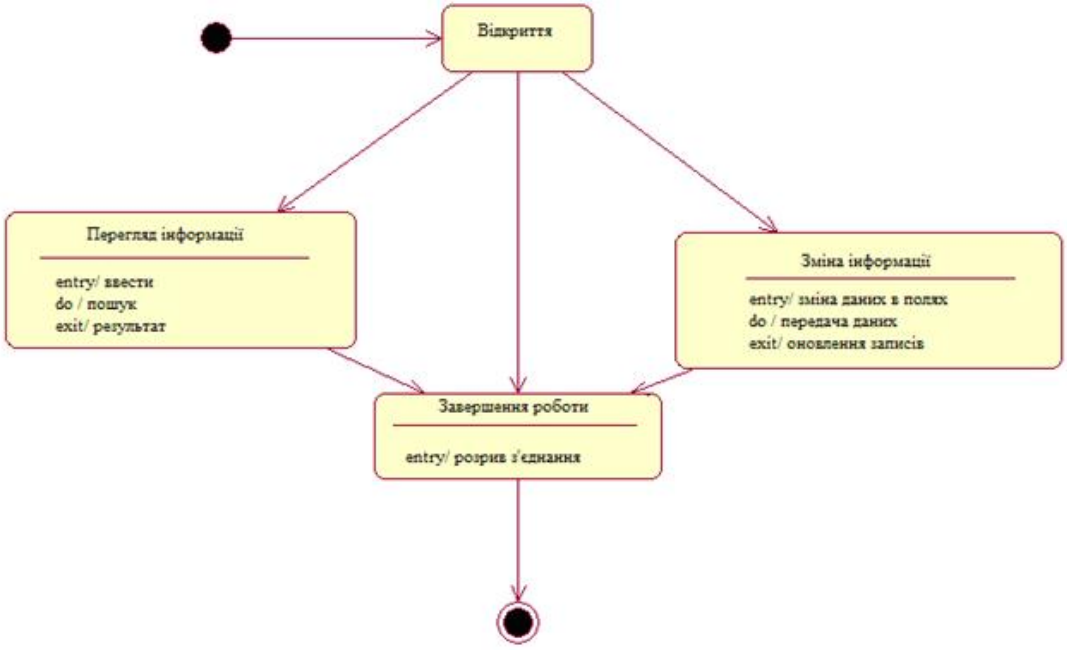

Рис. 2.13. Діаграма станів для класу «Progress»

Рис. 2.14 - Діаграма станів для класу «DBMS»

Діаграма компонентів включає заголовні файли та файли з вихідним кодом, для всіх класів, представлених на діаграмі класів [8].

Класи:

– DBMS;
– LoginForm;
– ProgressForm;
– ReportForm;
– MainWindow.

Готова діаграма компонентів зображена на рисунку 2.15.

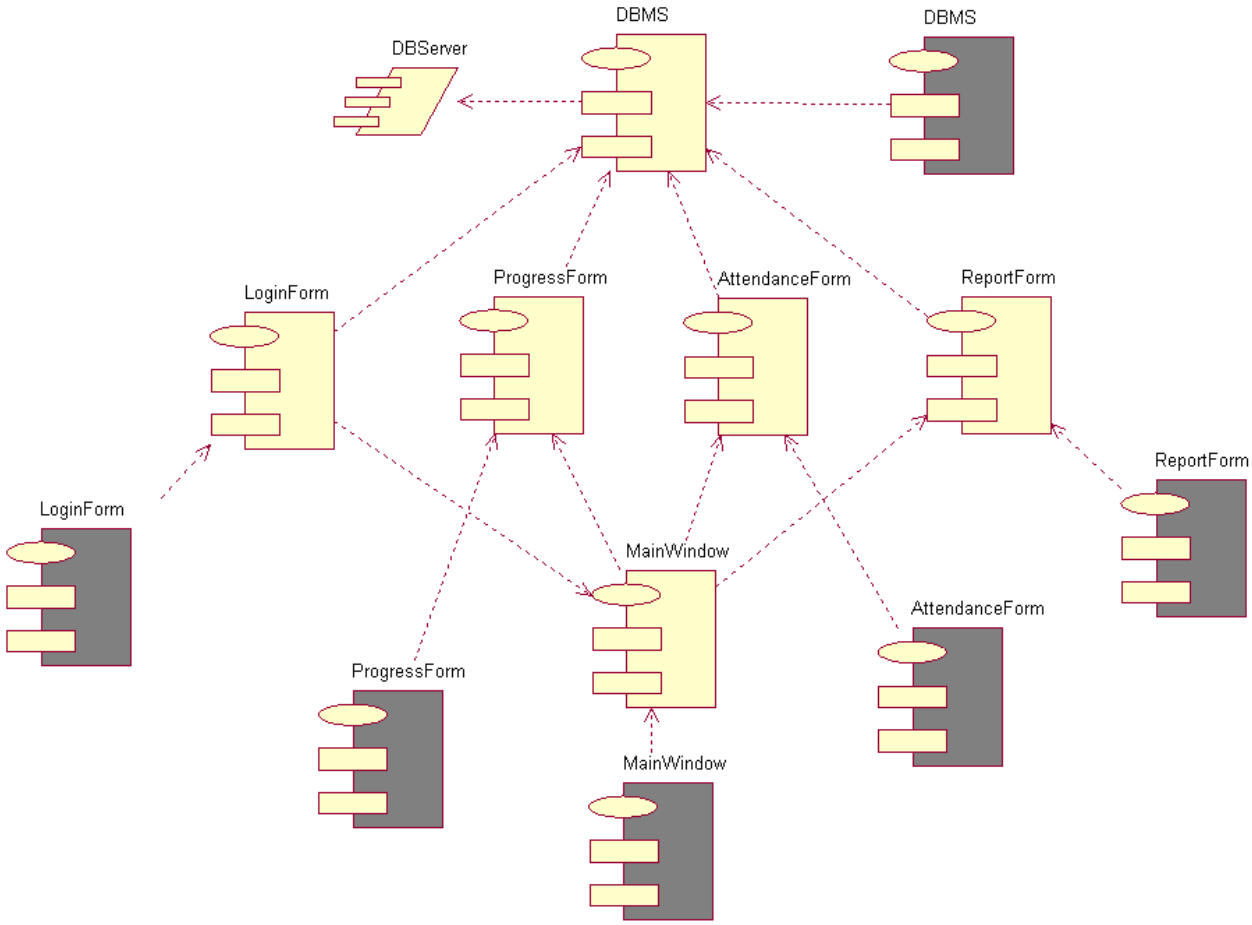

Рис. 2.15 - Діаграма компонентів для пакету «Облік споживання води»

2.3 Проектування структури бази даних

Одним з найбільш складних етапів проектування є розробка таблиць бази даних та збереження інформації, тому що результати, які повинна видавати система не завжди дають повну уяву про структуру таблиць.

При розробці, краще керуватися наступними принципами:

- Інформація в таблицях не повинна дублюватися. Коли визначена інформація зберігається тільки в одному місці, то немає необхідності у синхронізації цих даних, та забезпечить ефективність, та виключить можливість не збігу.
- Кожна таблиця повинна містити інформацію тільки на одну тему, у цьому випадку дані набагато легше обробляти, якщо вони утримуються в різних таблицях.
- Кожна таблиця бази даних повинна містити інформацію на окрему тему, а кожне поле таблиці — містити дані по темі таблиці.

При розробці треба враховувати:

- кожне поле повинне бути зв'язане з темою таблиці;
- не рекомендується включати в таблицю результати вираження;
- у таблиці повинна бути присутня уся необхідна інформація;
- інформацію варто розбивати на найменші логічні одиниці.

У зв’язку з великою кількістю інформації не має можливості відобразити та описати всі таблиці та зв’язки між ними. Тому виділимо процес розрахунку суми, яку необхідно сплатити фізичній особі за показаннями лічильників.

Модель «Сутність-зв'язок» (Entity-Relationship, або ER- модель) являє собою концептуальну модель даних. Модель даних являє собою набір концепцій, які описують структуру бази даних і пов'язані з нею транзакції модифікації та отримання даних. Основна мета розробки моделі полягає в створенні моделі користувацького сприйняття даних й узгодження великої кількості технічних аспектів пов'язаних із проектуванням бази даних.

Концептуальна модель даних не залежить від конкретної системи керування базою даних (СКБД) або апаратної платформи, що використовується для реалізації бази даних, Коли говорять про логічне проектування, уживають такі терміни, як сутність, зв'язок і атрибут.

Сутність це безліч однотипних об'єктів, називаних екземплярами, при цьому кожен екземпляр індивідуальний і відрізняється від всіх інших екземплярів.

Типи сутності можна класифікувати як сильні і як слабкі. Сильна сутність, це сутність яка може існувати незалежно від інших. Слабка сутність, це сутність існування якої залежить від сильної сутності. На ER- діаграмах сильні сутності позначаються прямокутником, а слабкі прямокутником з подвійним контуром [1].

Атрибут це характеристика сутності. Атрибут виражає одна закінчена і визначена властивість сутності. При проектуванні рекомендується створювати атомарні атрибути. Домен атрибута це набір значень який може бути привласнений атрибуту. Зв'язок - це логічне відношення між сутностями, що виражає деяке обмеження.

Ступінь участі сутності визначає чи залежить існування деякої сутності від участі у зв'язку іншої сутності. Ступінь участі може бути обов'язковою і необов'язковою. Кожен зв'язок зображується у вигляді ромбика із зазначеним на ньому ім'ям зв'язку. Значення зв'язку характеризує його тип. При створенні зв'язків між сутностями в дочірню сутність передаються атрибути, що складають первинний ключ у батьківській сутності. Ці атрибути утворюють у дочірній сутності зовнішній ключ. Потенційний ключ це атрибут або набір атрибутів які однозначно ідентифікують екземпляр сутності. Первинний ключ вибирається з потенційних ключів виходячи з гарантій унікальності його значень [2].

Проведений аналіз предметної області дозволяє побудувати діаграму «Сутність - зв'язок» яка зображена на рисунку 2.16

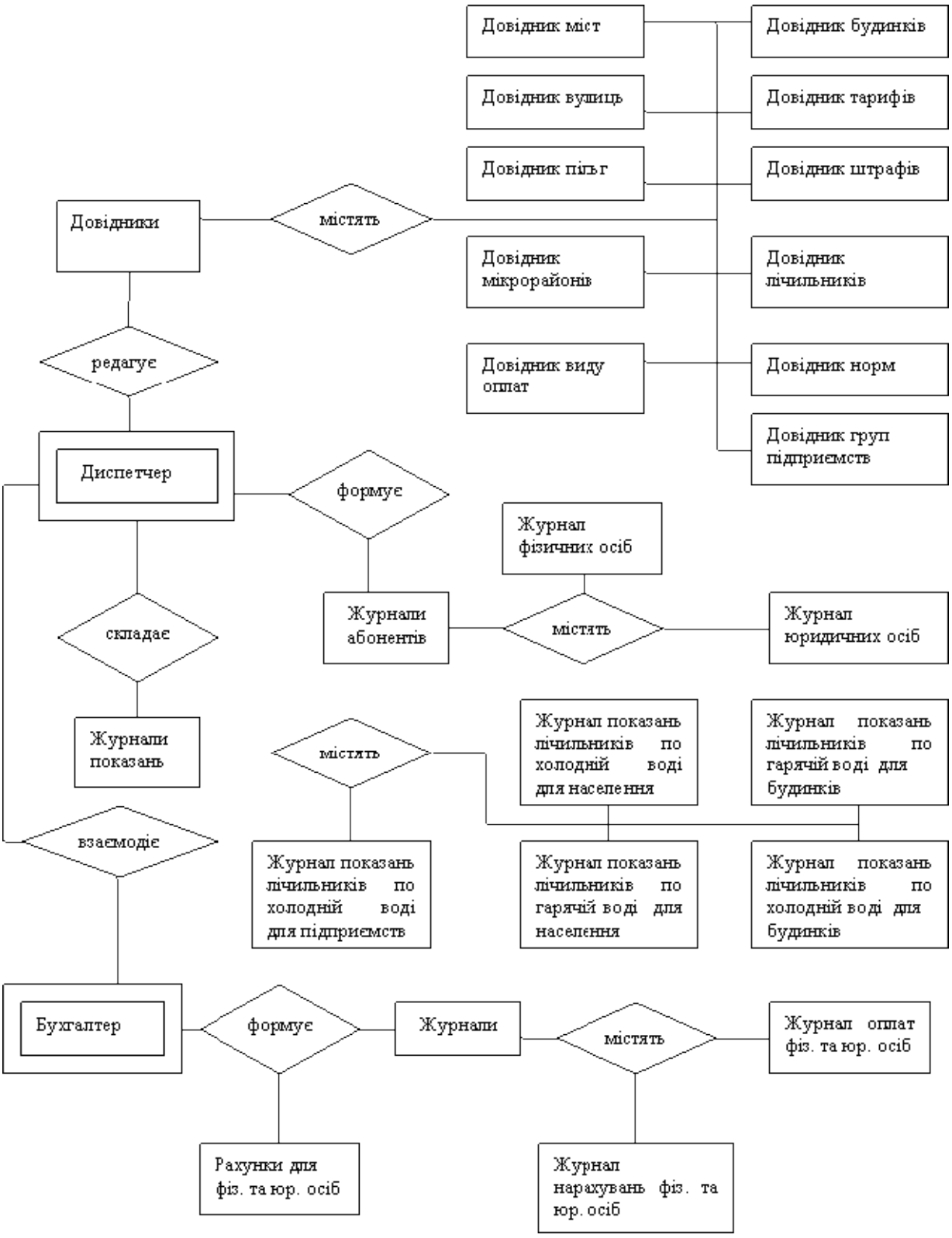

Рис. 2.16. Діаграма «Сутність-зв’язок»

Виділяємо наступні таблиці та атрибути (таблиця 2.1-2.11):

Таблиця 2.1

Опис структурних одиниць інформації вхідної і вихідної форми "Абоненти"

№	Повне найменування	Тип даних	Кількість символів
1	Ідентифікатор абонента	Цілочисельний	
2	Ідентифікатор будинку	Цілочисельний	
3	Ідентифікатор пільги	Цілочисельний	
4	Квартира	Символьний	10
5	Кількість мешканців	Цілочисельний	
6	Кількість соток	Символьний	255
7	Рахунок	Символьний	255
8	ПІП	Символьний	255
9	Телефон	Символьний	50
10	Дата договору	Дата	
11	Холодна вода	Символьний	20
12	Гаряча вода	Символьний	20
13	Каналізація	Символьний	20

Таблиця 2.1. представляє собою картку фізичної особи. Призначення атрибутів:

- Ідентифікатор абонента – унікальний код таблиці;
- Ідентифікатор будинку – ідентифікатор будинку, який є унікальним кодом таблиці Будинки;
- Ідентифікатор пільги – ідентифікатор пільги, який є унікальним кодом таблиці Пільги;
- Квартира – номер квартири мешканця;
- Кількість мешканців – кількість мешканців, які проживають за адресою;
- Кількість соток – площа земельної ділянки, призначеної для поливу;
- Рахунок – особистий рахунок абонента, який складається з коду вулиці, коду будинку, коду корпусу, коду квартири;
- ПІП – ПІБ абонента;
- Телефон – телефон абонента;

- Дата договору – дата складання договору абонента з Водоканалом;
- Холодна вода – ознака наявності холодної води у абонента;
- Гаряча вода – ознака наявності гарячої води у абонента;
- Каналізація – ознака наявності каналізації у абонента.

Таблиця 2.2

Опис структурних одиниць інформації вхідної і вихідної форми "Штрафи"

№	Повне найменування	Тип даних	Кількість символів
1	Ідентифікатор штрафу	Цілочисельний	
2	Ідентифікатор абонента	Цілочисельний	
3	Ідентифікатор виду штрафу	Цілочисельний	
4	Період початку	Дата	
5	Період завершення	Дата	
6	Дата	Дата	

Таблиця 2.2 містить відомості про штраф фізичної особи. Призначення атрибутів:

- Ідентифікатор штрафу – унікальний код таблиці;
- Ідентифікатор абонента – унікальний ідентифікатор таблиці Абоненти;
- Ідентифікатор виду штрафу – унікальний ідентифікатор таблиці Вид штрафу;
- Період початку – початок періоду, за який накладається штраф;
- Період завершення – закінчення періоду, за який накладається штраф;
- Дата – дата коли було накладено штраф.

Таблиця 2.3

Опис структурних одиниць інформації вхідної і вихідної форми "Журнал нарахувань"

№	Повне найменування	Тип даних	Кількість символів
1	Ідентифікатор журналу	Цілочисельний	
2	Номер документу	Символьний	25
3	Дата	Дата	
4	Початкова дата	Дата	
5	Кінцева дата	Дата	
6	Початок оплати	Дата	
7	Кінець оплати	Дата	
8	Сума боргу	Дробовий	
9	Загальна сума	Дробовий	
10	Сума за холодну воду	Дробовий	
11	Сума за гарячу воду	Дробовий	
12	Сума за каналізацію	Дробовий	
13	Сума за полив	Дробовий	
14	Примітка	Символьний	255
15	Проведення докумнета	Логічний	

Таблиція 2.3 створюється при складанні журналу нарахувань для фізичних осіб. Призначення атрибутів:

- Ідентифікатор журналу – унікальний код таблиці;
- Номер документу – номер документа розрахунків;
- Дата – дата створення документу;
- Початкова дата – дата початку нарахувань (для визначення періоду, за який сплачують);
- Кінцева дата – дата кінця нарахувань (для визначення періоду, за який сплачують);
- Початок оплати – початкова дата оплати (для визначення попередніх оплат при розрахунку боргів);
- Кінець оплати – кінцева дата оплати (для визначення попередніх оплат при розрахунку боргів);

- Сума боргу – сума загального боргу;
- Загальна сума – сума загальних нарахувань без врахування пільг;
- Сума за холодну воду – загальна сума нарахувань по холодній воді без врахування пільг;
- Сума за гарячу воду – загальна сума нарахувань по гарячій воді без врахування пільг;
- Сума за каналізацію – загальна сума нарахувань по каналізації без врахування пільг;
- Сума за полив – загальна сума за полив;
- Примітка – коментар;
- Проведення докумнета – ознака проведеного розрахунку.

Таблиця 2.4

Опис структурних одиниць інформації вхідної і вихідної форми "Нарахування"

№	Повне найменування	Тип даних	Кількість символів
1	Ідентифікатор нарахувань	Цілочисельний	
2	Ідентифікатор журналу	Цілочисельний	
3	Ідентифікатор рахунку абонента	Цілочисельний	
4	Куби за холодну воду	Дробовий	10
5	Куби за гарячу воду	Дробовий	10
6	Куби за каналізацію	Дробовий	10
7	Куби за полив	Дробовий	10
8	Сума пільг за воду	Дробовий	10
9	Сума пільг за каналізацію	Дробовий	10
10	Сума боргу за холодну воду	Дробовий	10
11	Сума боргу за гарячу воду	Дробовий	10
12	Сума боргу за каналізацію	Дробовий	10
13	Сума боргу за полив	Дробовий	10
14	Сума боргу	Дробовий	10
15	Нарахована сума	Дробовий	10
16	Дата	Дата	

Таблиця 2.4 потрібна для відображення даних після проведення розрахунків. Призначення атрибутів:

- Ідентифікатор нарахувань – унікальний код таблиці;
- Ідентифікатор журналу – унікальний ідентифікатор таблиці Журнал нарахувань;
- Ідентифікатор абонента – унікальний ідентифікатор таблиці Абонент;
- Куби за холодну воду – нараховані куби по холодній воді;
- Куби за гарячу воду – нараховані куби по гарячій воді;
- Куби за каналізацію – нараховані куби по каналізації;
- Куби за полив – нараховані куби за полив;
- Сума пільг за воду – нарахована сума до сплати з урахуванням пільг по воді;
- Сума пільг за каналізацію – нарахована сума до сплати з урахуванням пільг по каналізації;
- Сума боргу за холодну воду – нарахована сума боргу по холодній воді;
- Сума боргу за гарячу воду – нарахована сума боргу по гарячій воді;
- Сума боргу за каналізацію – нарахована сума боргу по каналізації;
- Сума боргу за полив – нарахована сума боргу за полив;
- Сума боргу – нарахована загальна сума боргу;
- Нарахована сума – нарахована сума до сплати;
- Дата – місяць, за який ведуться нарахування.

Таблиця 2.5

Опис структурних одиниць інформації вхідної і вихідної форми "Пільги"

№	Повне найменування	Тип даних	Кількість символів
1	Ідентифікатор пільги	Цілочисельний	
2	Найменування	Символьний	255
3	Проценти	Дробовий	10
4	Дата введення	Дата	
5	Код пільги	Символьний	255
6	Коротка назва	Символьний	255

Таблиця 2.5 представляє відомості про пільги. Призначення атрибутів:

- Ідентифікатор пільги – унікальний код таблиці;
- Найменування – найменування категорії пільг;
- Проценти – процентна знижка по пільзі;
- Дата введення – дата введення пільги;
- Коротка назва – коротке найменування пільги;
- Код пільги – код пільги.

Таблиця 2.6

Опис структурних одиниць інформації вхідної і вихідної форми " Пільговики"

№	Повне найменування	Тип даних	Кількість символів
1	Ідентифікатор пільговика	Цілочисельний	
2	Ідентифікатор абонента	Цілочисельний	
3	Ідентифікатор спорідненості	Цілочисельний	
4	ПІП	Символьний	255

Таблиця 2.6 призначена для зберігання відомостей про пільговиків. Призначення атрибутів:

- Ідентифікатор пільговика – унікальний код таблиці;
- Ідентифікатор абонента – унікальний ідентифікатор таблиці Абонент;
- Ідентифікатор спорідненості – унікальний ідентифікатор таблиці Степінь спорідненості;
- ПІБ – ПІБ пільговика.

Таблиця 2.7

Опис структурних одиниць інформації вхідної і вихідної форми "Вулиця"

№	Повне найменування	Тип даних	Кількість символів
1	Ідентифікатор вулиці	Цілочисельний	
2	Код вулиці	Символьний	255
3	Назва	Символьний	255

Таблиця 2.7 - Призначення атрибутів:

- Ідентифікатор вулиці – унікальний код таблиці;
- Код вулиці – код вулиці (для складання особистого рахунку абонента);
- Назва – назва вулиці.

Таблиця 2.8

Опис структурних одиниць інформації вхідної і вихідної форми "Будинки"

№	Повне найменування	Тип даних	Кількість символів
1	Ідентифікатор будинка	Цілочисельний	
2	Ідентифікатор міста	Цілочисельний	
3	Ідентифікатор вулиці	Цілочисельний	
4	Ідентифікатор літери	Цілочисельний	
5	Ідентифікатор мікрорайону	Цілочисельний	
6	Номер	Символьний	10
7	Дріб	Символьний	10
8	Корпусу	Символьний	10
9	Ознака приватного сектору	Логічний	

Таблиця 2.8 - Призначення атрибутів:

- Ідентифікатор будинка – унікальний код таблиці;
- Ідентифікатор міста – унікальний ідентифікатор таблиці Міста;
- Ідентифікатор вулиці – унікальний ідентифікатор таблиці Вулиці;
- Ідентифікатор літери – унікальний ідентифікатор таблиці Літери;
- Ідентифікатор мікрорайону – унікальний ідентифікатор таблиці Мікрорайонів;
- Номер – номер будинку;
- Дріб – дріб будинку;

- Корпус – корпус будинку;
- Ознака приватного сектору – ознака того, що будинок розташований в приватному секторі.

Таблиця 2.9

Опис структурних одиниць інформації вхідної і вихідної форми "Оплата абонентів"

№	Повне найменування	Тип даних	Кількість символів
1	Ідентифікатор оплати	Цілочисельний	
2	Ідентифікатор абонента	Цілочисельний	
3	Ідентифікатор виду платежу	Цілочисельний	
4	Початкові показники	Дробовий	10
5	Кінцеві показники	Дробовий	10
6	Дата оплати	Дата	
7	Сума оплати	Дробовий	10

Таблиця 2.9 показує вже існуючі на даний момент оплати абонентів. Призначення атрибутів:

- Ідентифікатор оплати – унікальний код таблиці;
- Ідентифікатор абонента – унікальний ідентифікатор таблиці Абонент;
- Ідентифікатор виду платежу – унікальний ідентифікатор таблиці Види платежу;
- Початкові показники – початкові показання лічильника (окремо для холодної води, гарячої води, каналізації, полива);
- Кінцеві показники – кінцеві показання лічильника (окремо для холодної води, гарячої води, каналізації, полива);
- Дата оплати – дата сплати рахунку;
- Сума оплати – суми сплати.

Таблиця 2.10

Опис структурних одиниць інформації вхідної і вихідної форми "Показники холодної води"

№	Повне найменування	Тип даних	Кількість символів
1	Ідентифікатор показника	Цілочисельний	
2	Ідентифікатор абонента	Цілочисельний	
3	Початкові показники	Дробовий	10
4	Кінцеві показники	Дробовий	10
5	Загальні показники	Дробовий	10
6	Початкова дата	Дата	
7	Кінцева дата	Дата	

Таблиця 2.10 Показники холодної води дозволяє вести облік спожитої кількості холодної води та надає інформацію для розрахунків. Призначення атрибутів:

- Ідентифікатор показника – унікальний код таблиці;
- Ідентифікатор абонента – унікальний ідентифікатор таблиці Абонент;
- Початкові показники – початкові показання лічильників холодної води;
- Кінцеві показники – кінцеві показання лічильників холодної води;
- Загальні показники –кількість нарахованих кубів холодної води;
- Початкова дата – дата початку періоду;
- Кінцева дата – дата закінчення періоду.

Таблиця 2.11

Опис структурних одиниць інформації вхідної і вихідної форми "Показники гарячої води"

№	Повне найменування	Тип даних	Кількість символів
1	Ідентифікатор показника	Цілочисельний	
2	Ідентифікатор абонента	Цілочисельний	
3	Початкові показники	Дробовий	10
4	Кінцеві показники	Дробовий	10
5	Загальні показники	Дробовий	10
6	Початкова дата	Дата	
7	Кінцева дата	Дата	

Таблиця 2.10 Показники гарячої води дозволяє вести облік спожитої кількості холодної води та надає інформацію для розрахунків. Призначення атрибутів:

- Ідентифікатор показника – унікальний код таблиці;
- Ідентифікатор абонента – унікальний ідентифікатор таблиці Абонент;
- Початкові показники – початкові показання лічильників гарячої води;
- Кінцеві показники – кінцеві показання лічильників гарячої води;
- Загальні показники –кількість нарахованих кубів гарячої води;
- Початкова дата – дата початку періоду;
- Кінцева дата – дата закінчення періоду.

Після розподілу даних по таблицях і визначення полів, необхідно вибрати схему для зв'язку даних у різних таблицях. Для цього потрібно визначити ключові поля і зв'язки між таблицями.

При створенні таблиць, у кожну нову таблицю включається поле, що зв'язує нову й стару таблиці. Ці сполучні поля називаються зовнішніми ключами. У добре спроектованій базі даних використання зовнішніх ключів забезпечує ефективність використання додатку. У процесі проектування потрібно уважно стежити за створенням зовнішніх ключів. Заключний етап

логічного проектування бази даних полягає у визначенні зв'язків між таблицями. Зв'язки первинних ключів, що задають при створенні таблиць із зовнішніми ключами використовуються для об'єднання даних з декількох таблиць.

У більшості випадків, як уже описувалося вище, таблиці зв'язуються між собою відношенням «один до багатьох», набагато рідше «один до одного» і «багато до багатьох». Якщо в базі даних існує зв'язок між таблицями типу «багато до багатьох», то необхідно створити таблицю перетинання, за допомогою якої один зв'язок «багато до багатьох» буде зведено до двох зв'язків типу «багато до одного». У дійсній базі даних всі таблиці будуть зв'язуватися між собою відношенням типу «один до багатьох».

Визначимо зв'язки між спроектованими таблицями й існуючими в базі. Таблиця Абонент пов’язана з таблицями Штрафи абонентів, Нарахування, Пільговики, Показники холодної води, Показники гарячої води, Оплата абонентів за допомогою первинного ключа Ідентифікатор абонента, тип зв’язку буде «один до багатьох».

З таблицею Будинки та Пільги вона зв’язана за допомогою зовнішнього ключа Ідентифікатор будинку, тип зв’язку буде «багато до одного». З таблицею Пільги вона зв’язана за допомогою зовнішнього ключа Ідентифікатор пільги, тип зв’язку буде «багато до одного».

Таблиця Журнал нарахувань зв’язана з таблицею Нарахування за допомогою первинного ключа Ідентифікатор нарахувань, тип зв’язку буде «один до багатьох».

Таблиця Будинки зв’язана з таблицею Вулиці за допомогою зовнішнього ключа Ідентифікатор вулиці, тип зв’язку «один до багатьох». На рисунку 2.17 представлена логічна модель даних для системи обліку споживання води.

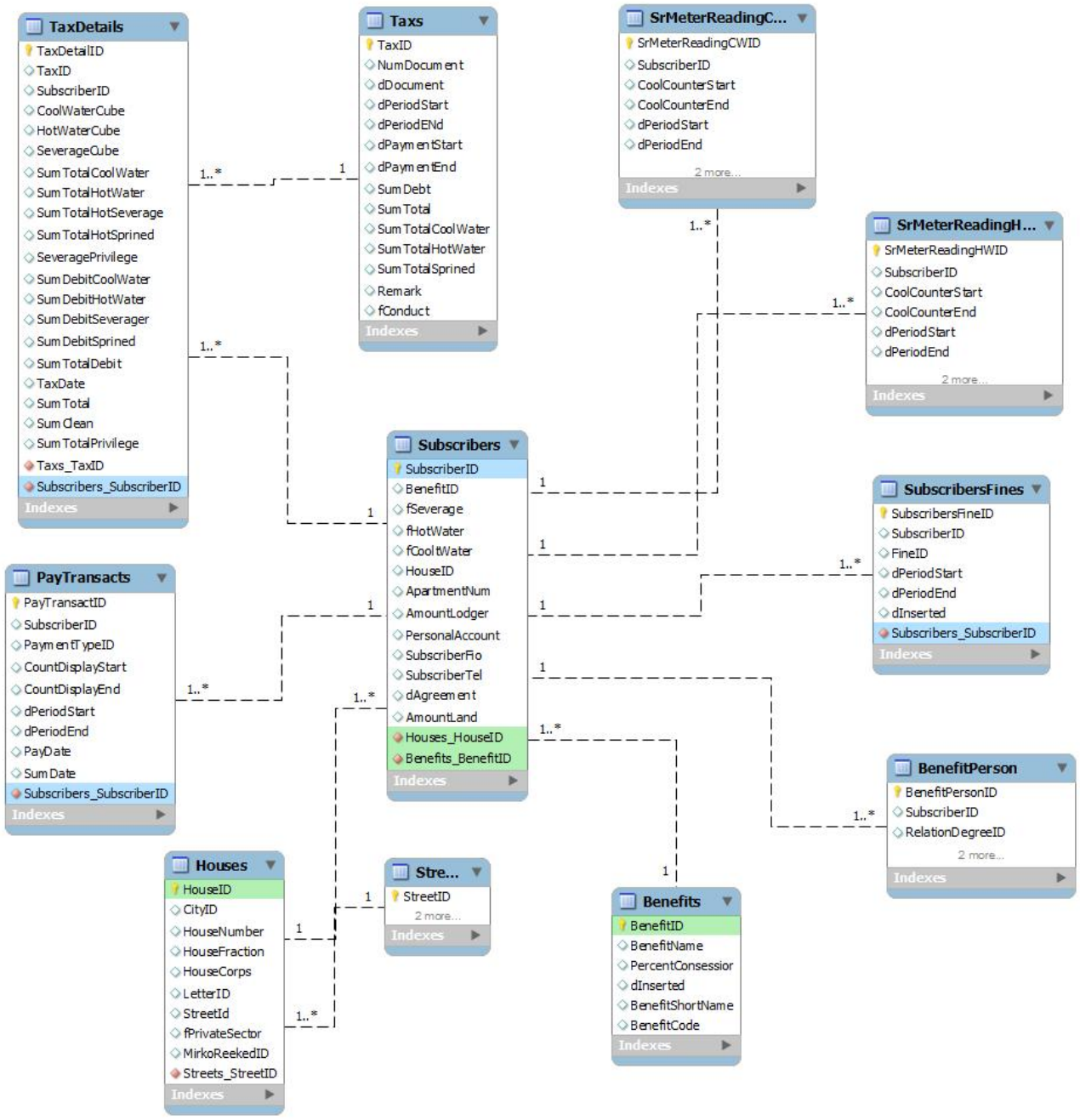

Рис. 2.17. Логічна модель даних системи споживання води

Висновки до розділу 2

У цьому розділі розроблено архітектуру програмного додатку, що дозволить краще зрозуміти функції основних його частин. Створено та описано структурну схему, основними компонентами якої є: рівень клієнта, рівень бізнес-логіки та рівень даних. Описано функціональну структуру системи та її основних елементів – модулів обробки даних. Визначено основні елементи бази даних та встановлено зв’язки між ними. Спроектовано структуру бази даних.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ОБЛІКУ СПОЖИВАННЯ ВОДИ

3.1. Програмна реалізація системи

Для реалізації системи обілку споживання води вибраного мову програмування Java. Java — програма компілюється не зразу в машинні команди, не в команди якогось конкретного процесора, а в команди так званої віртуальної машини Java (JVM, Java Virtual Machine), що представлено на рисунку 3.1. Віртуальна машина Java — це сукупність команд разом з системою їх виконання. Віртуальна машина Java повністю стекова, так що не вимагається складна адресація комірок памяті і велика кількість регістрів. Тому команди JVM короткі, більшість з них має довжину 1 байт, звідси команди JVM називають байткодами (bytecodes), хоча є команди довжиною 2 і 3 байти. Згідно статистичних дослідженнь середня довжина команди складає 1,8 байта. Повне описання команд і всієї архітектури JVM міститься в специфікації віртуальної машини Java(VMS, Virtual Machine Specification).

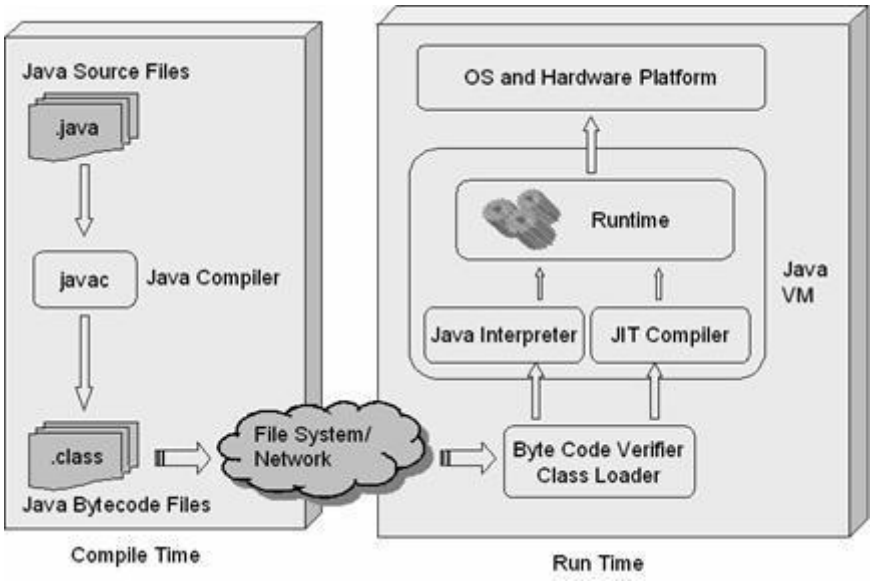

Рис. 3.1. Схема виконання програм на Java

Друга особливість Java — всі стандартні функції, що викликаються в програмі, підключаються до неї тільки на етапі виконання, а не включаються в байт-коди. Відбувається динамічне компонування (dynamic binding). Це теж сильно зменшує об’єм скомпільованої програми.

Отже, на першому етапі програма, написана на мові Java, переводиться компілятором в байт-коди. Ця компіляція не залежить від типу якого-небудь конкретного процесора і архітектури деякого конкретного компютера. Вона може бути виконана зразу ж після написання програми. Байт-коди записуються в одному або декількох файлах, можуть зберігатися у зовнішній пам’яті або передаватися по мережі. Це особливо зручно дякуючи невеликому розміру файлів з байт-кодами. Потім одержані в результаті компіляции байт-коди можна виконувати на будь-якому комп’ютері, котрий має систему реалізації JVM. При цьому не має значення ні тип процесора, ні архітектура комп’ютера.

Інтерпретація байт-кодів і динамічне компонування значно сповільнюють виконання програм. Це не має значення в тих ситуаціях, коли байт-коди передаються по мережі, мережа все рівно повільніша любої інтерпретації, але в інших ситуаціях вимагається потужний і швидкий компютер. Тому постійно йде вдосконалення інтерпретаторів в сторону збільшення швидкості інтерпретації. Разроблені JIT-компілятори (Just-In-Time), запамятовуючі уже інтерпретовані частки кода в машинних командах процесора і просто виконуючі ці участки при повторному зверненні, наприклад, в циклах. Це значно збільшує швидкість обчислень, що повторяються. Фірма SUN розробила цілу технологію Hot-Spot і включає її в свою віртуальну машину Java. Але, звичайно, найбільшу швидкість може дати тільки спеціалізований процесор.

Фірма SUN Microsystems випустила мікропроцесори PicoJava, що працюють на системі команд JVM, Ці процесори безпосередньо виконують байт-коди. Але при виконанні програм Java на інших процесорах вимагається ще інтерпретація команд JVM в команди конкретного процесора, а значить,

потрібна програма-інтерпретатор, причому для кожного типу процесорів, і для кожної архітектури компютера треба написати свій інтерпретатор.

Це завдання уже виішено практично для всіх комп'ютерних платформ. На них реалізовані віртуальні машини Java, а для найбільш розповсюджених платформ існує декілька реалізацій JVM різних фірм. Все більше операційнних систем і систем управління базами даних включають реалізацію JVM в своє ядро. Створена і спеціальна операційна система JavaOS, яка застосовується в електронних пристроях. В більшості браузерів вбудована віртуальна машина Java для виконання аплетів.

Крім реалізації JVM для виконання байт-кодів на комп'ютері ще потрібно мати набір функцій, які викликаються із байт-кодів і динамічно компонуються з байт-кодами. Цей набір оформляється у вигляді бібліотеки класів Java, яка складається з одного або декількох пакетів. Кожна функція може бути записана байт-кодами, але, оскільки вона буде зберігатися на конкретному комп'ютері, її можна записати прямо в системі команд цього комп'ютера, уникнувши тим самим інтерпретації байт-кодів. Такі функції називають "рідними" методами (native methods). Застосування "рідних" методів прискорює виконання програми.

Фірма SUN Microsystems — творець технології Java — безкоштовно розповсюджує набір необхідних програмних інструментів для повного циклу роботи с цією мовою програмування: компіляції, інтерпретації, налаштовування, який включає і багату бібліотеку класів, під назвою JDK (Java Development Kit). Є набори інструментальних програм і інших фірм.

Набір програм і класів JDK містить:

- компілятор javac із вихідного тексту в байт-коди;
- інтерпретатор java, який містить реалізацію JVM;
- полегшений інтерпретатор jre (в останніх версіях відсутній);
- програму перегляду аплетів appletviewer, що заміняє браузер;
- налаштовувач jdt;
- дизассемблер javap;

- програму архівації і стиснення jar;
- програму збору документації javadoc;
- програму javah генерації заголовкових файлів мови C;
- програму javakey додавання електронного підпису;
- програму native2ascii, яка перетворює бінарні файли в текстові;
- програми rmic і rmiregistry для роботи з віддаленими об’єктами;
- програму serialver, яка визначає номер версії класу;
- бібліотеки і заголовкові файли "рідних" методів;
- бібліотеку класів Java API (Application Programming Interface).

В попередні версії JDK включались і налаштовувальні варіанти виконуваних програм: javac_g, java_g і т.д. Компанія SUN Microsystems постійно розвиває і обновляє JDK, кожний рік появляються нові версії.

Набір програм і пакетів класів JRE містить все необхідне для виконання байт-кодів, в тому числі інтерпретатор java (в попередніх версіях полегшений інтерпретатор jre) і бібліотеку класів. Це частина JDK, яка не містить компілятори, налаштовувачі і інші засоби розробки. Якраз JRE або його аналог інших фірм міститься в браузерах, що вміють виконувати програми на Java, в операційних системах і системах управління базами даних.

JDK містить вихідні тексти більшості своїх програм, написані на Java. Це дуже зручно. Завжди можна з точністю дізнатись, як працює той чи інший метод обробки інформації із JDK, проглянувши вихідний код даного метода.

JDK являється частиною якогось інтегрованого середовища програмування, наприклад JBuilder , то деякі з перечислених вище папок можуть знаходитися в іншому місці.

NetBeans IDE — вільне інтегроване середовище розробки (IDE) для мов програмування Java, JavaFX, C/C++, PHP, JavaScript, Python, Groovy. Середовище може бути встановлене і для підтримки окремих мов, і у повній конфігурації. Середовище розробки NetBeans за умовчанням підтримує розробку для платформ J2SE і J2EE (рисунок 3.2).

Проект NetBeans IDE підтримувався фірмою Sun Microsystems і після придбання Sun — Oracle, проте розробка NetBeans ведеться незалежно співтовариством розробників (NetBeans Community) і компанією NetBeans.Org.

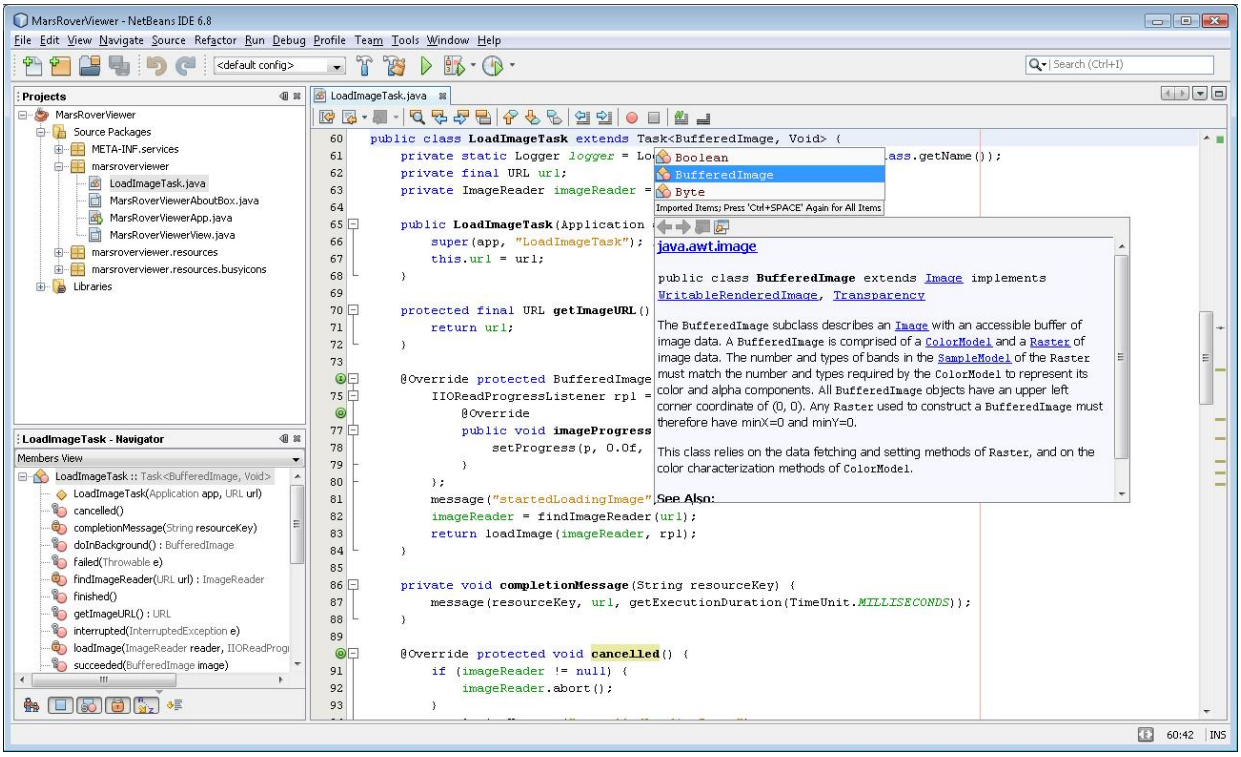

Рис. 3.2. Середовище розробки NetBeans IDE

За якістю і можливостям останні версії NetBeans IDE змагається з найкращим інтегрованими середовищами розробки для мови Java, підтримуючи рефакторинг, профілювання, виділення синтаксичних конструкцій кольором, автодоповнення мовних конструкцій на льоту, шаблони коду та інше. NetBeans IDE доступна для платформ Microsoft Windows, GNU/Linux, FreeBSD, і Solaris (як SPARC, так x86). Для інших платформ доступна можливість зібрати NetBeans самостійно.

3.2. Програмна реалізація бази даних

Фізичне проектування це створення опису реалізації бази даних на запам'ятовувальних пристроях з визначенням структур зберігання й методів

доступу, використовуваних для організації ефективної обробки даних. Фізичне проектування є фазою процесу створення проекту бази даних при виконуванні якої проектувальник приймає рішення щодо способів реалізації бази даних. Приступаючи до фізичного проектування бази даних необхідно, насамперед, вибрати конкретну СУБД. Взагалі, основною метою фізичного проектування бази даних є опис способу фізичної реалізації логічного проекту бази даних. У випадку реляційної моделі під цим мається на увазі наступне:

- створення набору реляційних таблиць та обмежень для них на основі інформації представленої в глобальній логічній моделі даних;
- визначення конкретних структур зберігання даних і методів доступу до них, що забезпечують оптимальну продуктивність системи з базою даних;
- розробка засобів захисту створюваної системи від несанкціонованого доступу.

Для програмної реалізації бази даних необхідна СУБД з відкритим вихідним кодом, яка підтримує обсяги даних до 1 Тб та достатню продуктивність. Із рішень з відкритим вихідним кодом, що відповіжають вказаним вимогам, найбільш розповсюдженими є MySQL.

Система керування реляційними базами даних MySQL була розроблена компанією «TcX» 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. У 1999 році MySQL обігнала mSQL за популярністю. Остання версія mSQL- 3.8 - була випущена 9 червня 2006.

MySQL виникла як спроба застосувати mSQL до власних розробок компанії: таблицям, для яких використовувалися ISAM — підпрограми низького рівня. У результаті був вироблений новий SQL-інтерфейс, але API-інтерфейс залишився в спадок від mSQL.

З часом MySQL все розширювалася і зараз вона — одна з найпоширеніших систем керування базами даних. Вона використовується, в

першу чергу, для створення динамічних веб-сторінок, оскільки має впевнену підтримку з боку різноманітних мов програмування.

Станом на квітень 2015 року, MySQL пропонувала версію MySQL 5.6 в двох різних варіантах: з відкритим вихідним кодом MySQL Community Server і комерційний Server Enterprise. Вони мають спільний програмний код і включають в себе серед іншого наступні можливості:

- Крос-платформна підтримка.
- Збережувані процедури та функції.
- Тригери.
- Курсори.
- Оновлювані подання (представлення).
- Інформаційна схема (так званий системний словник, що містить метадані).
- Підтримка SSL.
- Кешування запитів.
- Вкладені запити SELECT.
- Підтримка реплікації.
- Повноцінна підтримка Юнікоду (UTF-8 і UCS2).
- Сегментування таблиць.

MySQL являється компактним багатопоточним сервером баз даних та характеризується великою швидкістю, стійкістю і простотою використання.

Ця система вважається гарним рішенням для малих і середніх додатків. Вихідний код сервера компілюється на безлічі платформ. Найбільш повно можливості сервера виявляються в UNIX-системах, де є підтримка багатопоточності, що підвищує продуктивність системи в цілому. Для некомерційного використання MySQL є безкоштовним.

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.

Недоліками сервера MySQL залишається не реалізована підтримка транзакцій, замість якої пропонується використовувати LOCK/UNLOCK TABLE, а також відсутність підтримки для зовнішніх (foreign) ключів, тригерів, збережених процедур та представлень (VIEW). Та для розробки малих та середніх інформаційних систем, призначених для використання в межах робочих груп ці недоліки є фактично невідчутними.

У текстовому режимі робота з базою даних виглядає просто як введення команд у командний рядок, а результати вибірок повертаються у вигляді своєрідних таблиць, поля в яких налазять один на одного, якщо дані не вміщаються на екрані.

Розглянемо детальніше процедуру реалізації відношень спроектованої бази дани SoftWoter в СУБД MySQL. На рисунку 3.3 представлено реалізацію відношення Subscribers, яке містить інформацію про абонентів.

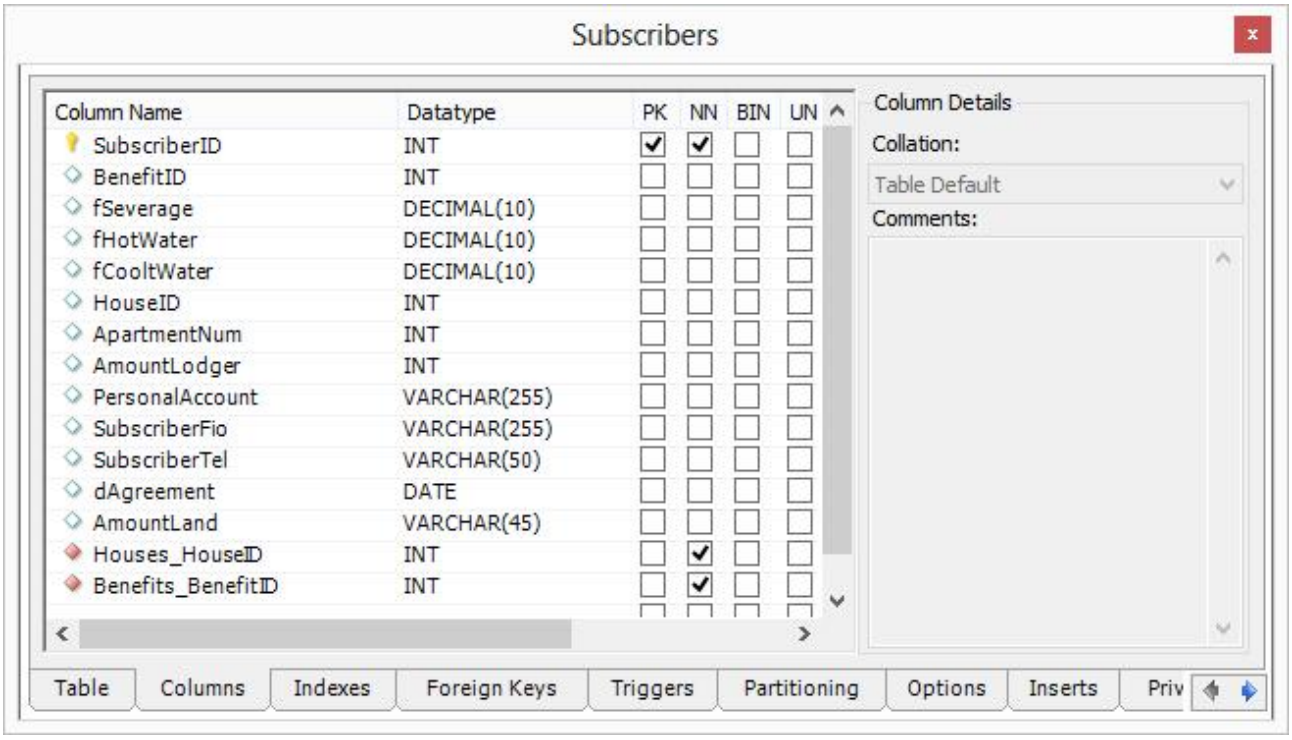

Рис. 3.3. Реалізація відношення Subscribers

На рисунку 3.4 представлено реалізацію відношення SubscribersFines, яке містить усю інформацію про штрафи обонетів.

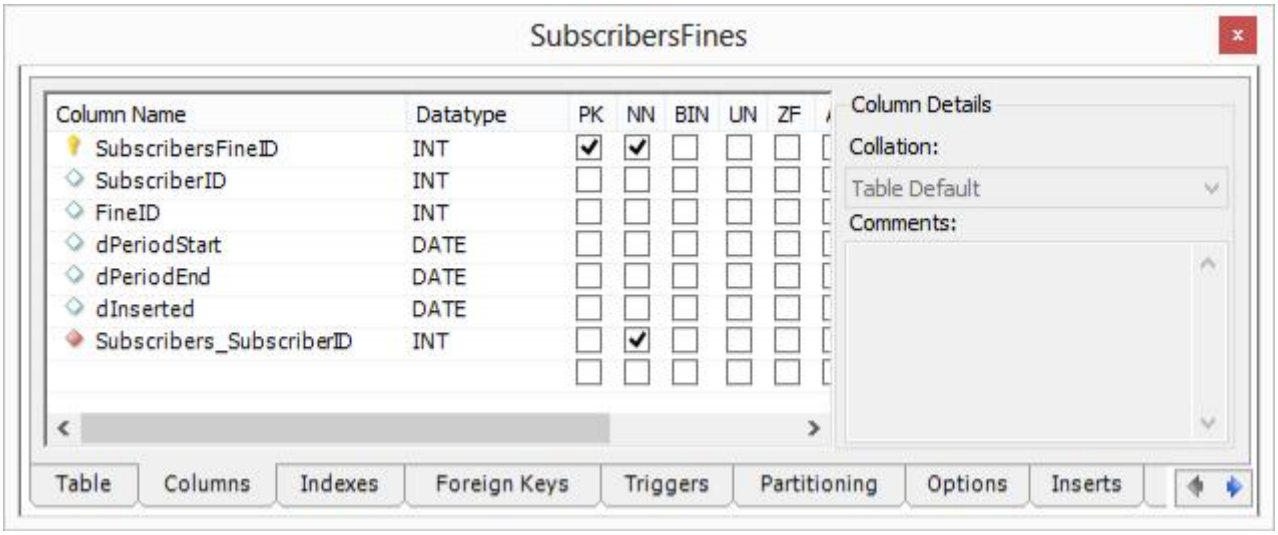

Рис. 3.4. Реалізація відношення SubscribersFines

На рисунку 3.5 – реалізація відношення Taxs, в якому зберегіються дані журлану нарахувань за спожиту воду.

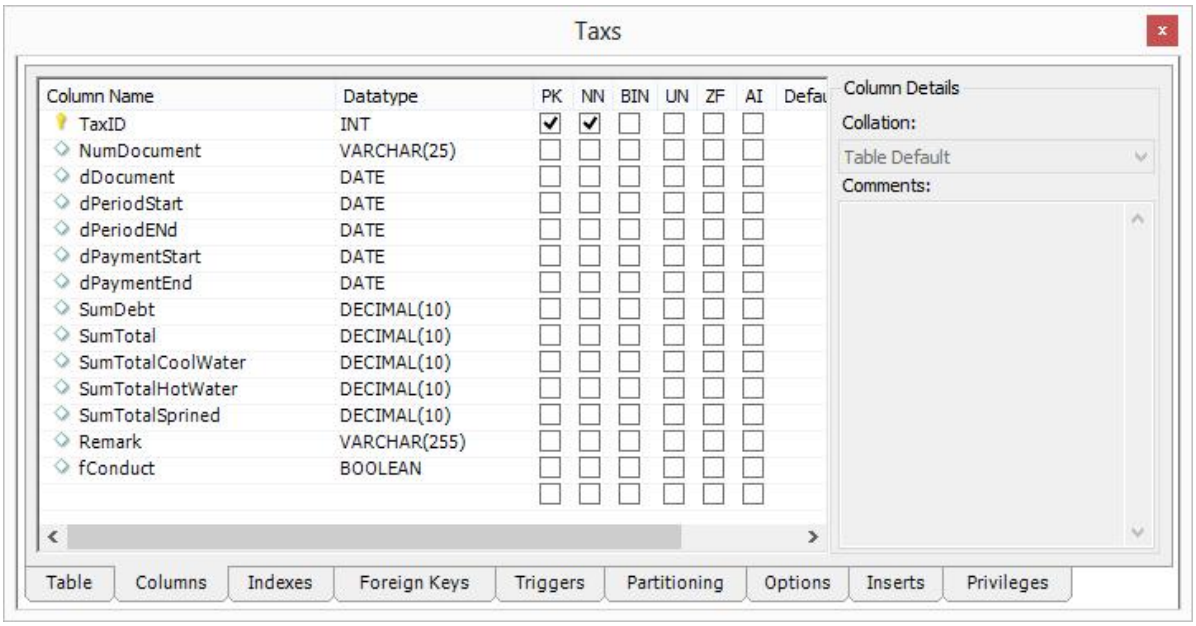

Рис. 3.5. Реалізація відношення Taxs

На рисунку 3.6 представлено реалізацію відношення TaxDetails, в якому зберегаються дані щодо деталізації здійснених нарахувань.

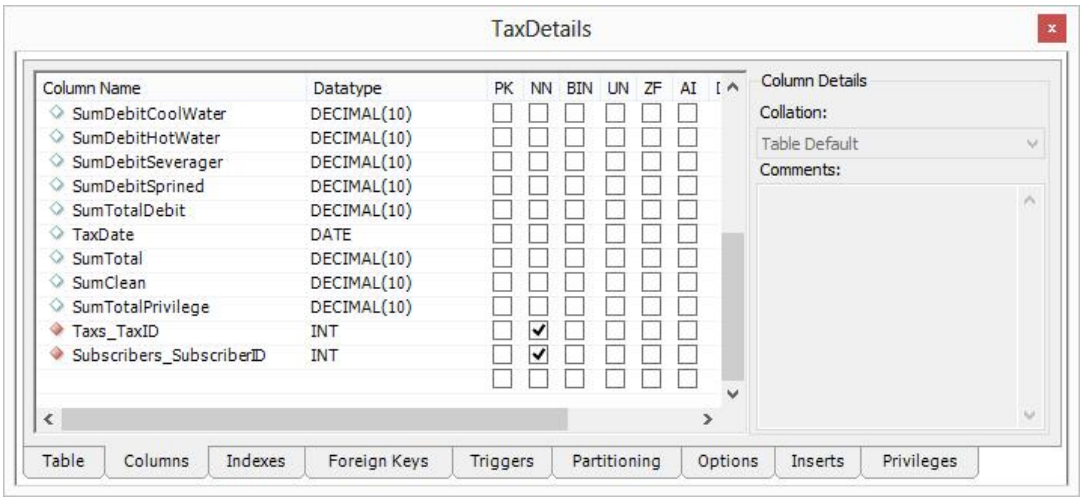

Рис. 3.6. Реалізація відношення TaxDetails

На рисунку 3.7 наведено структуру відношення Benefits, в якому зберігаються дані про основні та додаткові пільги абонентів.

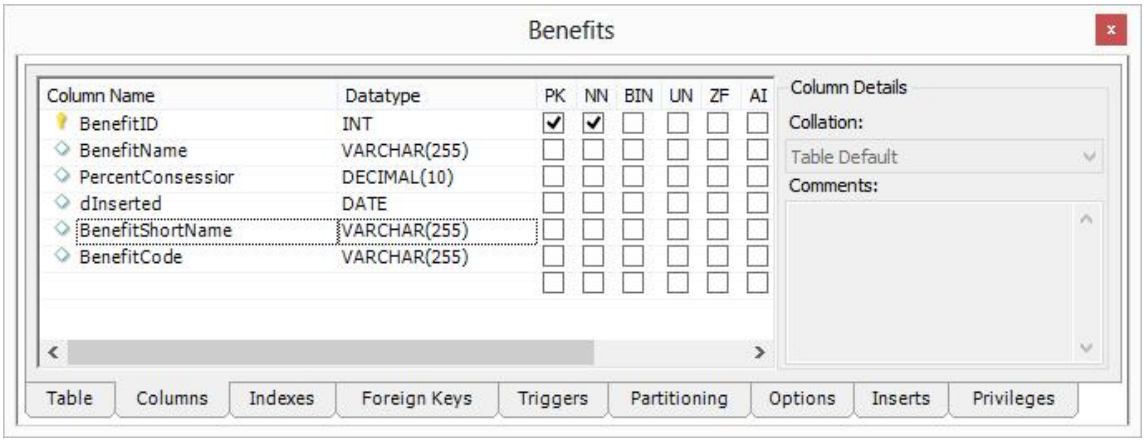

Рис. 3.7. Реалізація відношення Benefits

На рисунку 3.8 представлено реалізацію відношення BenefitPerson, яке містить інформацію про абонента, якому призначена пільга, її розмір.

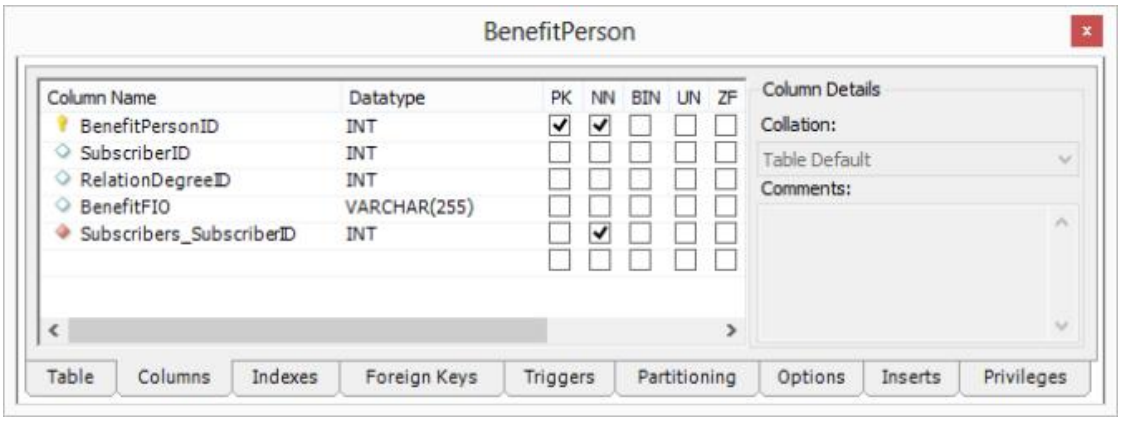

Рис. 3.8. Реалізація відношення BenefitPerson

На рисунку 3.9 представлено реалізацію відношення Streets, яке містить інформацію про вулиці.

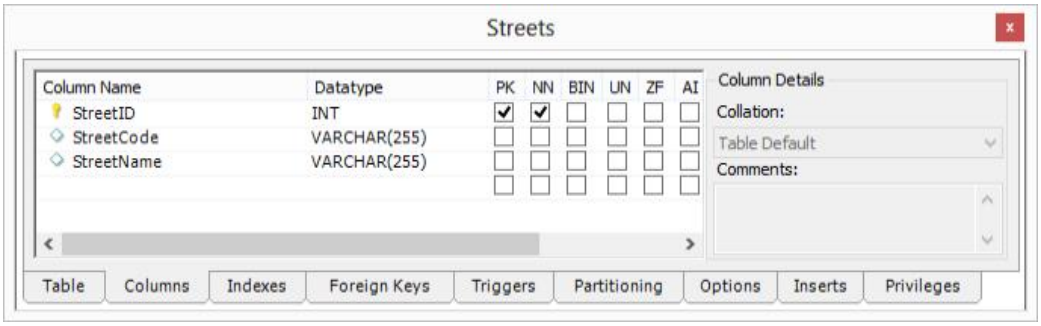

Рис. 3.9. Реалізація відношення Streets

На рисунку 3.10 – реалізація відношення Houses, в якому зберегіється інформація про будинки.

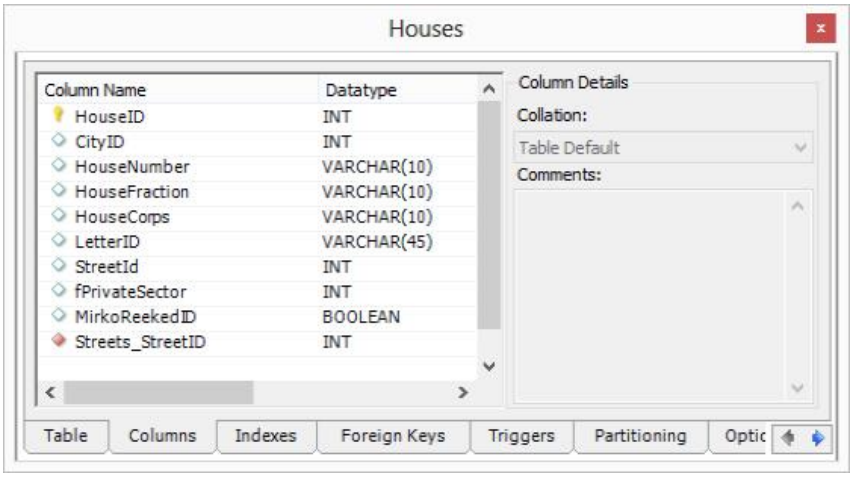

Рис. 3.10. Реалізація відношень Houses

На рисунку 3.11 – реалізація відношення PayTransacts, в якому зберігається інформація про історію оплат обонентів за воду.

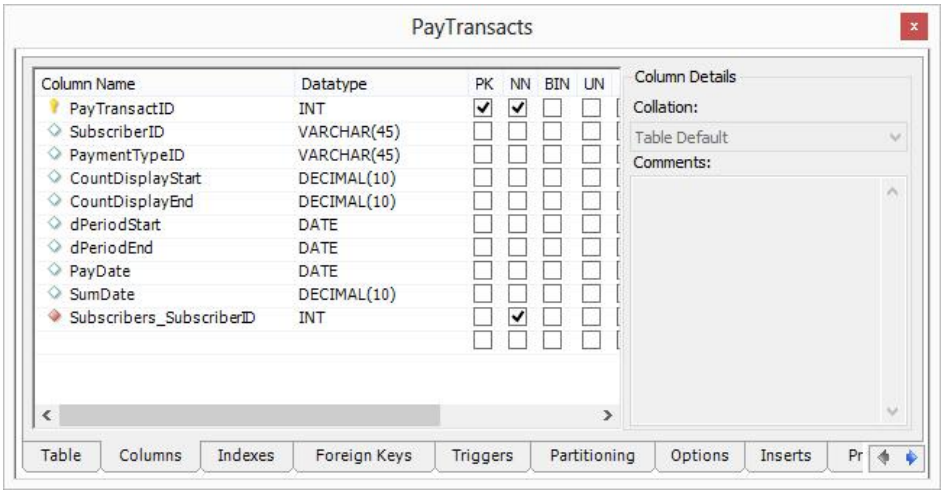

Рис. 3.11. Реалізація відношень PayTransacts

На рисунку 3.12 – реалізація відношення SrMeterReadingCWs, в якому зберегіється інформація про показники лічильників холодної води.

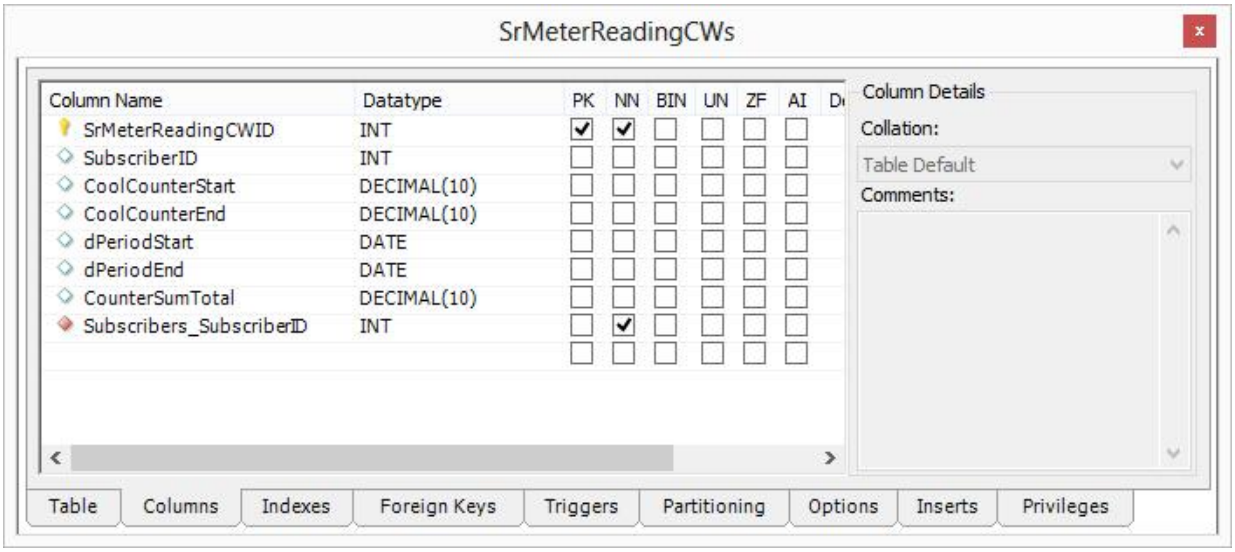

Рис. 3.12. Реалізація відношень SrMeterReadingCWs

На рисунку 3.13 – реалізація відношення SrMeterReadingHWs, в якому зберегіється інформація про показники лічильників гарячої води.

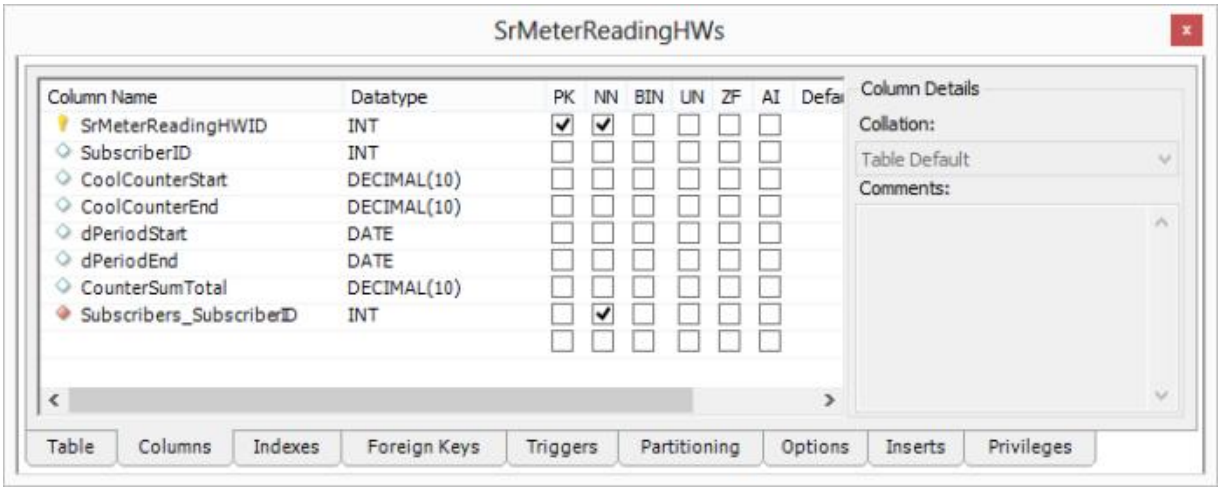

Рис. 3.13. Реалізація відношень SrMeterReadingHWs

Підсумкову ER діаграму представлено на рисунку 3.14. З даного рисунка видно представлення зв’язків міє таблицями БД, та їх конкретну реалізацію в середовищі MySQL.

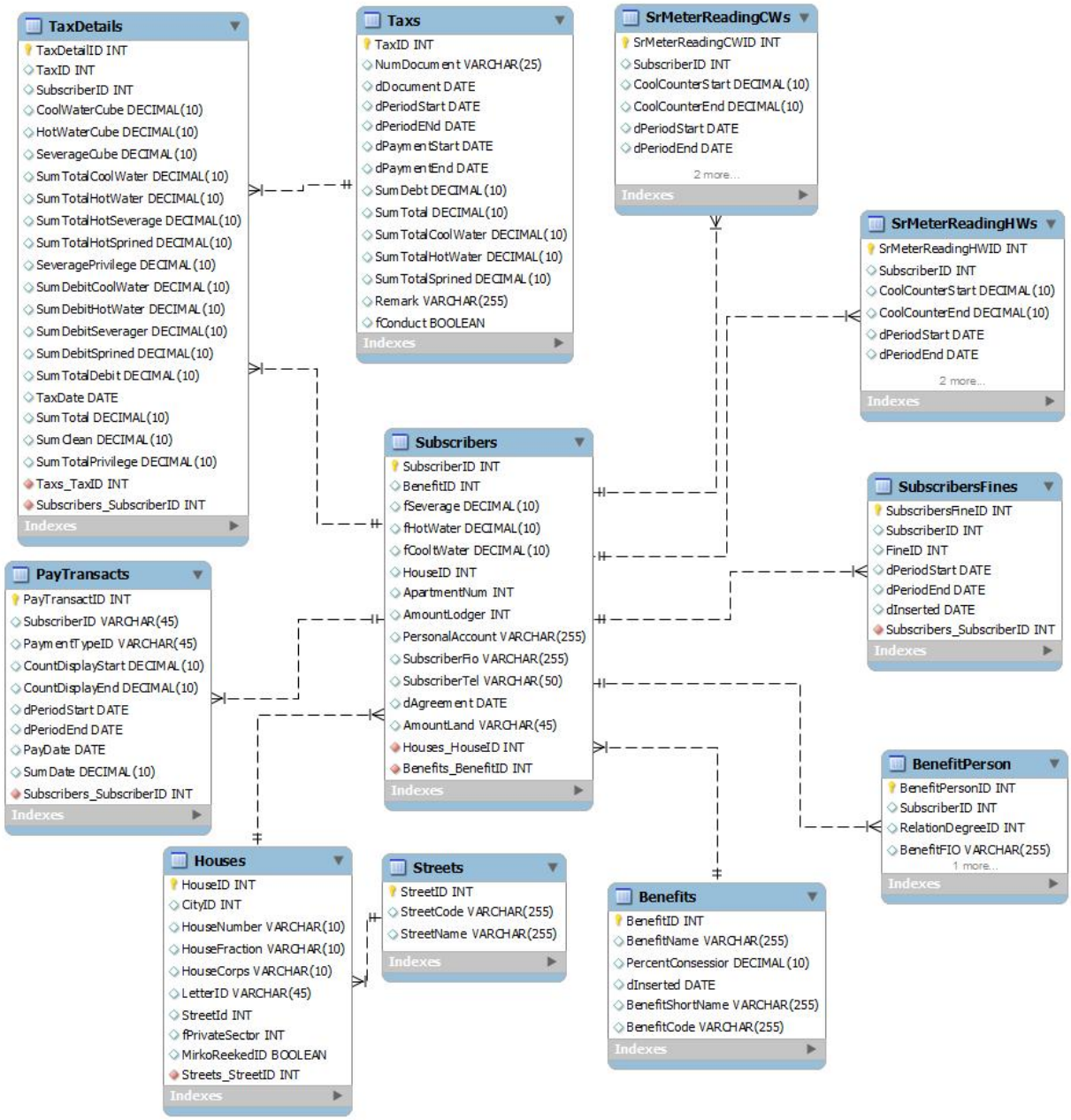

Рис. 3.14. Діаграма «сутність-зв'язок» бази даних

У додатку Б дипломної роботи представлено DDL statement бази даних SoftWoter.

Висновки до розділу 3

У даному озділі обґрунтовано технологію, мову програмування та розроблено програмну систему. Обґрунтовано засоби розробки бази даних та створено програмну реалізацію бази даних програмної системи за допомогою механізму збережених процедур.

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ СИСТЕМИ ОБЛІКУ СПОЖИВАННЯ ВОДИ ТЕПЛОМЕРЕЖЕЮ

4.1. Тестування системи

Тестування програмного забезпечення - це процес, що використовується для виміру якості розроблюваного програмного забезпечення. Зазвичай, поняття якості обмежується такими поняттями, як коректність, повнота, безпечність, але може містити більше технічних вимог, які описані в стандарті ISO 9126. Тестування - це процес технічного дослідження, який виконується на вимогу замовників, і призначений для вияву інформації про якість продукту відносно контексту, в якому він має використовуватись. До цього процесу входить виконання програми з метою знайдення помилок [10].

Досить часто компонентні додатки використовують кілька компонентів, які, в свою чергу, використовують ресурси на сервері і працюють в контексті сервера. Наприклад, набір класів Java може використовувати джерело даних для популярної РСУБД (реляційної системи управління базою даних), сконфігурованої на сервері додатків. Ці об’єкти Java, таким чином, використовують під час роботи джерело даних в контексті сервера.

Тестування компонентів, які використовують ресурси сервера, трохи складне, оскільки ці компоненти знаходяться в контексті сервера. Подібні компоненти можна протестувати на рівні модуля одним із таких способів.

Модульне тестування компонента з локальним ресурсом. Наприклад, припустимо, що компонент використовує джерело даних РСУБД, налаштований на сервері додатків. Тоді тестування цього компонента може включати в себе налаштування його для локальної РСУБД або будь-який іншої файлової системи. Хоча цей підхід і допомагає, якщо необхідно швидко протестувати компонент без особливого зв’язку з серверними ресурсами, у

нього є один недолік. Він не відображає реальне середовище, в якій знаходиться компонент.

Модульне тестування компонента шляхом створення поверх нього додатку-оболонки з подальшим розгортанням цієї оболонки на сервері разом з компонентом. Однак недоліком цього підходячи є те, що розробка тестового додатка-оболонки вимагатиме значних зусиль.

Модульне тестування компонента шляхом запуску тестового коду в тому ж контексті, що і у компонента. Наприклад, тестовий код можна запустити в самому серверному контексті, де буде працювати компонент.

Розглянемо процедуру тестування системи обліку споживання води, яка реалізована на джерелі даних компонент, і яку можна розгорнути на сервері додатків за допомогою середовища JUnit і платформи Rational Application Developer.

Здійснимо процедуру тестування функції аутентифікації користувача. Система встановлює справжність користувача, який вводить своє ім'я і пароль. Список призначених для користувача імен і паролів вже завантажений в БД, і додаток перевіряє відповідність цих даних кожен раз, коли користувач намагається зареєструватися в додатку.

Тепер, в цьому сценарії, компонент аутентифікації повинен взяти в якості вхідних параметрів ім'я користувача і пароль, щоб переглянути їх відповідність даним в БД, і повернути відповідне повідомлення про підтвердження особи користувача. Таким чином, компонент аутентифікації може використовувати такі класи (рисунок 4.1):

- клас User, який абстрагує користувача в додатку;
- клас DataSource, який забезпечує зв'язок між БД і додатком;
- клас Authenticator, який насправді перевіряє відповідність введених даних користувача інформації в БД.

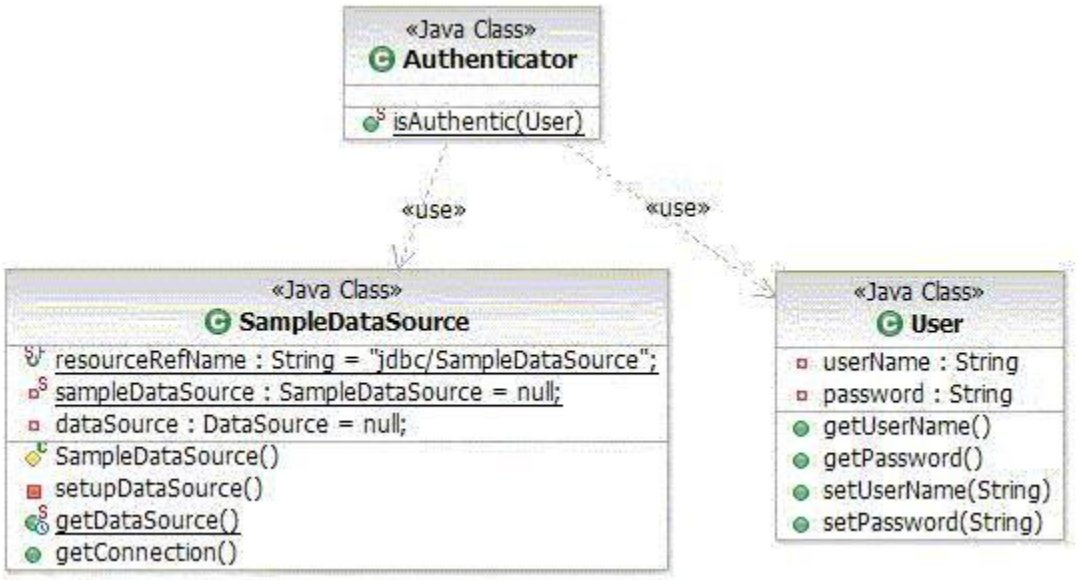

Рис. 4.1. Фрагмент діаграми класів для аутентифікації в системі

Розглянемо класи User.java, SampleDataSource.java та Authenticator.java. Далі приведено лістинг вказаних класів.

Клас User.java:

```
package com.ibm.datasource.sample;
public class User {
  private String userName;
  private String password;
  public User() {}
  public User(String name, String pwd) {
    userName = name;
    password = pwd;
  }
  public String getUserName() { return userName; }
  public String getPassword() { return password; }
  public void setUserName(String userName) { this.userName = userName; }
  public void setPassword(String password) { this.password = password; }
}
```

Клас Authenticator.java:

```
package com.ibm.datasource.sample;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
import com.ibm.datasource.sample.db.SampleDataSource;
public class Authenticator {
  public static boolean isAuthentic(User user) {
    boolean authentic = false;
    try {
      SampleDataSource sampleDataSource = SampleDataSource.getDataSource();
```

```
                                Connection                      connection                      =
sampleDataSource.getConnection();
            Statement statement = connection.createStatement();
            String sql = "SELECT count(*) AS count FROM SAMPLEUSER WHERE username='"
                +   user.getUserName()   +   "'   and   password='"   +   user.getPassword()   +
"'";
            ResultSet resultSet = statement.executeQuery(sql);

            if(resultSet != null) {
              if(resultSet.next()) {
                int count = resultSet.getInt("count");
                if(count > 0) { authentic = true; }
              }
            }
          } catch(Exception e) { e.printStackTrace(); }
          return authentic;
        }
      }
```

Клас SampleDataSource.java:

```
package com.ibm.datasource.sample.db;
import java.sql.Connection;
import java.sql.SQLException;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;
public class SampleDataSource {
  public static final String resourceRefName="jdbc/SampleDataSource";
  private static SampleDataSource sampleDataSource = null;
  private DataSource dataSource = null;
  protected SampleDataSource() {}
  private void setupDataSource() throws SQLException {
    try {
      InitialContext ctx = new InitialContext();
      dataSource = (DataSource)ctx.lookup(resourceRefName);
    } catch(NamingException e) { e.printStackTrace(); }
  }
  public static synchronized SampleDataSource getDataSource() throws SQLException {
    if(sampleDataSource == null) {
      try {
        sampleDataSource = new SampleDataSource();
        sampleDataSource.setupDataSource();
      } catch(SQLException e) { e.printStackTrace(); }
    }
    return sampleDataSource;
  }
  public Connection getConnection() throws SQLException       {
    return dataSource.getConnection();
  }
}
```

Розглянемо реалізацію контрольних прикладів JUnit для підсистеми аутентифікації. Для цього створюємо пакет com.ibm.datasource.sample.test в

папці appClientModule. На рисунку 4.2 представлено контрольний приклад JUnit.

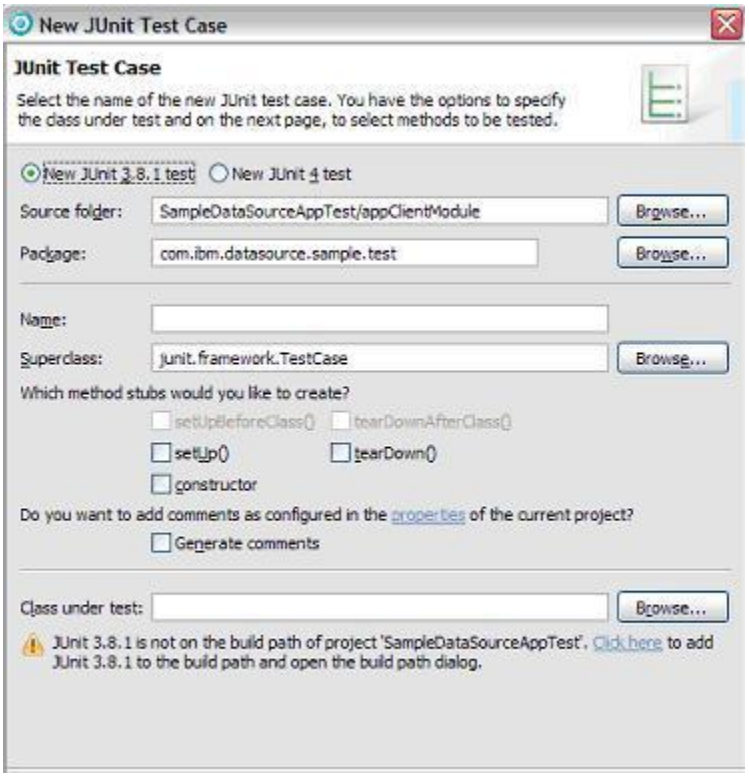

Рис. 4.2. Контрольний приклад JUnit

У нижній частині діалогового вікна, показаного на рисунку 4.2, необхідно додати бібліотеки JUnit, клацнувши по посиланню Click here додаємо бібліотеки JUnit в проект для клієнта програми і вводимо UserAuthenticationTest в поле Name. Це буде назва контрольного прикладу. Додаємо наступний код в тестовий приклад - клас UserAuthenticationTest:

```
package com.ibm.datasource.sample.test;
import com.ibm.datasource.sample.Authenticator;
import com.ibm.datasource.sample.User;
import junit.framework.TestCase;
public class UserAuthenticationTest extends TestCase {
  public void testAuthenticUser() {
    User user = new User("Mohan","mohan");
    assertTrue(Authenticator.isAuthentic(user));
  }
  public void testNonAuthenticUser() {
                  // пароль для користувача хибний
    User user = new User("Mohan","");
    assertFalse(Authenticator.isAuthentic(user));
  }
}
```

Тестовий клас містить два тести: один для аутентифікації правильного користувача, а інший - для аутентифікації користувача з неправильно зазначеними даними.

Для створення набору тестів клацаємо правою кнопкою миші на пакеті com.ibm.datasource.sample.test і вибераємо пункт меню New> Other> JUnitTest Suite (рисунок 4.3).

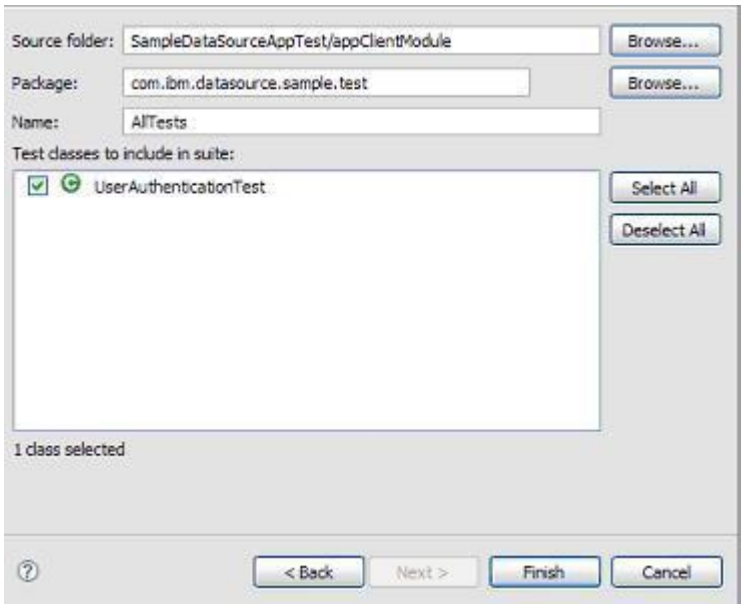

Рис. 4.3. Набір тестів JUnit

Клас UserAllTestJava:

```
package com.ibm.datasource.sample.test;
public class AllTests {
  public static void main(String[] av) {
    junit.textui.TestRunner.run(UserAuthenticationTest.class);
  }
}
```

На рисунку 4.4 представлено задання нової конфігурації.

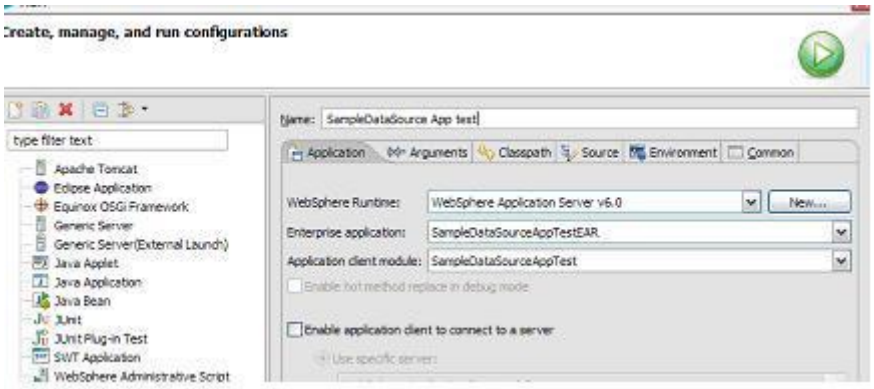

Рис. 4.4. Створення та запуск конфігурації в JUnit

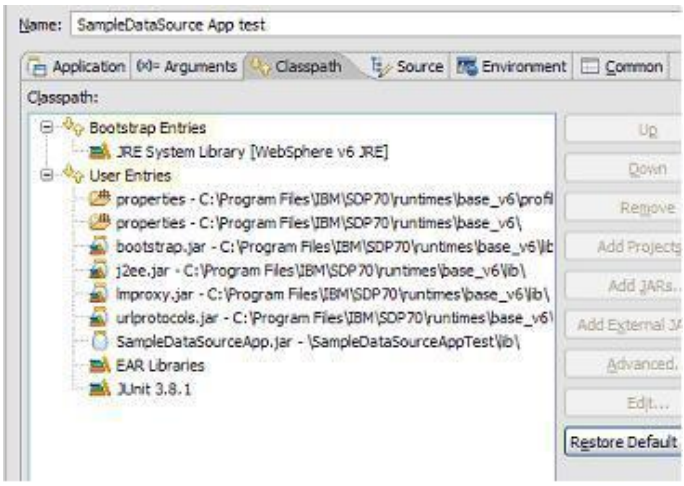

Рис. 4.5. Задання змінних оточення ClassPath

Після налаштування усіх конфігурацій робимо запуск усіх контрольних тестів за допомого команди Run. На рисунку 4.6 представлено результати тестування для розглянутої підсистеми аутенифікації.

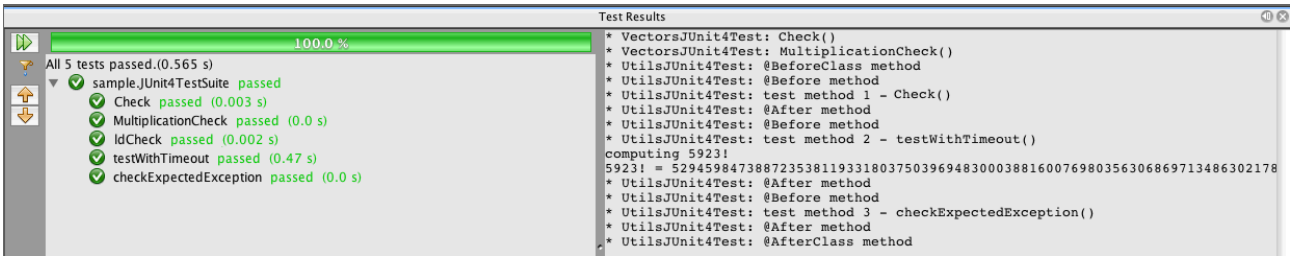

Рис. 4.6. Результати тестування

У прикладі на рисунку 4.6 у вікні відображаються результати змішаного набору тестів. Цей набір тестів включає набір тестів JUnit 4 і один тестовий клас JUnit 3. У наборі тестів тестові класи UtilsJUnit3Test.java і JUnit4TestSuite.java були виконані як один тест, і результати виведені на екран в лівій панелі як результати одного тесту. Дані в правій панелі являють собою результати виконання тестів окремо.

4.2. Розгортання програмного продукту

Для розгортання системи складемо список необхідних вимог до програмного забезпечення:

- Операційна система Microsoft Windows 7/8;
- MySQL;
- Eclipse IDE for Java EE Developers;
- Драйвера JDBC-MySQL;

Вимоги до апаратних ресурсів визначатимуться кількістю користувачів системи. Для невеликої кількості (до 100 одночасно) користувачів достатньо наступної конфігурації для апаратних ресурсів:

- процесор з тактовою частотою не менше 2 Ghz;
- оперативна пам’ять 4 Gb;
- об’єм жорсткого диску має бути не менше 512 Gb; необхідна пропускна здатність залежить від кількості користувачів.

Розглянемо процедуру розгортання проекту, яка складається з двох частин:

- розгортання Java додатку;
- налатування доступу до сервера баз даних MySQL;

Адаптер сервера WTP Eclipse - це інструмент для розгортання та тестування ресурсів Java EE на сервері WebSphere Application Server Community Edition. Перед тим, як приступити до розгортання ресурсів Java EE, слід визначити новий сервер і середовище виконання сервера. Після створення або імпорту проекту Java EE, в інтегроване середовище розробки (IDE) Eclipse, вказуємо середовище виконання сервера WebSphere Application Server Community Edition в якості цільової середовища. Ця дія додає бібліотеки класів сервера в шлях компонування проекту.

Ресурси, розгорнуті за допомогою функцій Eclipse, рекомендується видаляти і повторно розгортати також за допомогою функцій Eclipse. Наприклад, якщо розгорнути ресурс в Eclipse і потім видалити його за допомогою Web-консолі або команди deploy, то Eclipse буде вважати, що ресурс розгорнутий. Для усунення такої неполадки слід видалити ресурси, які були опубліковані на сервері і вилучені за межами середовища Eclipse.

Для того щоб розгорнути ресурси Java EE на локальному сервері, виконаємо нступні дії:

- У проекції Java EE вибераємо панель Проект і правою кнопкою миші на потрібному проекті Java EE. Вибераємо Виконати як, Запустити на сервері. На панелі Запустити на сервері, якщо сервер встановлено, вибераємо опцію Вибрати існуючий сервер і вибрати потрібний сервер. Якщо сервер Community Edition не заданий, вибераємо опцію Задати новий сервер вручну щоб задати новий сервер. Натискаємо кнопку Готово. Адаптер сервера WTP незабаром розгорне ресурси Java EE. Якщо сервер не запущений, адаптер сервера WTP запустить його і розгорне ресурс Java EE після ініціалізації сервера.

При необхідності ресурс, опублікований на сервері, можна видалити за допомогою опції Додати/Видалити проект. Якщо просто видалити ресурс без видалення пов'язаного проекту, ресурс залишиться розгорнутим на сервері. На панелі Сервер клацаємо правою кнопкою миші на сервері, в якому вимагається розгорнути ресурс. У контекстному меню вибераємо Додати/Видалити проекти.

Процедура розгортання ресурсу Java EE на віддаленому сервері аналогічна розглянутої вище, однак перед її виконанням рекомендується звернути увагу на додаткові особливості.

Для визначення віддаленого сервера необхідно спершу задати локальний сервер і потім в атрибуті імені хоста вказати ім'я хоста віддаленого сервера. Це обов'язкова процедура, оскільки середовище Eclipse використовує бібліотеки класів локального сервера.

За допомогою Eclipse не можна запустити, зупинити або перезапустити віддалений сервер. За допомогою Eclipse не можна запустити віддалений сервер в режимі налагодження. Як правило, внаслідок такого обмеження зручно вибрати підхід, що передбачає розробку і налагодження ресурсів Java EE на локальному сервері з подальшим переходом на віддалений сервер, призначений для інтеграції ресурсів кількох розробників. Відома неполадка Apache Geronimo може викликати непередбачену виняткову ситуацію java.net.UnknownHostExc.

Незалежно від того, яким чином вказано ім'я хоста цільового сервера (навіть у тому випадку, якщо вказано повне ім'я або IP-адреса), цільовий сервер повертає неповне ім'я хоста. Іншими словами він повертає клієнту своє коротке ім'я. Відповідно з цим ім'ям клієнт запускає операцію передачі файлу. Якщо мережа не може перетворити це ім’я, видається повідомлення про виняткову ситуацію. Якщо команда ping дозволяє звернутися до цільової системі за неповним іменем хоста, віддалене розгортання повинно працювати правильним чином. Опис способу обходу цієї неполадки наведено в розділі Усунення неполадок.

Якщо локальна система та цільової сервер розділені брандмауером, необхідно додатково налаштувати передачу запитів HTTP і RMI між ними. Після встановлення сервера за замовчуванням застосовуються порт HTTP 8080 і порт RMI 1099. Якщо в конфігурації сервера вказані інші порти, необхідно додатково налаштувати брандмауер для застосування цих портів. Під час виклику команди deploy повинен виконуватися віддалений сервер. Перед розгортанням або оновленням ресурсу Java EE пов'язані файли передаються по мережі і зберігаються в якості тимчасових файлів.

Налаштування драйвера JDBC-MySQL. Драйвер jdbc треба встановлювати та підключати додатково. Наприклад, для MySQL його можна взяти на http://dev.mysql.com/downloads/mirror.php?id=412737. Для роботи драйвера JDBC необхідно налаштувати OpenOffice.org/LibreOffice для роботи з Java. Отриманий файл-архів треба розпакувати в каталог. Потім підключити його до OpenOffice.org/LibreOffice. Сервіс-Параметри -OpenOffice.org - Java - Шлях класу-Додати архів і вибрати файл драйвера (mysql-connector-java-5.1.25-bin.jar).

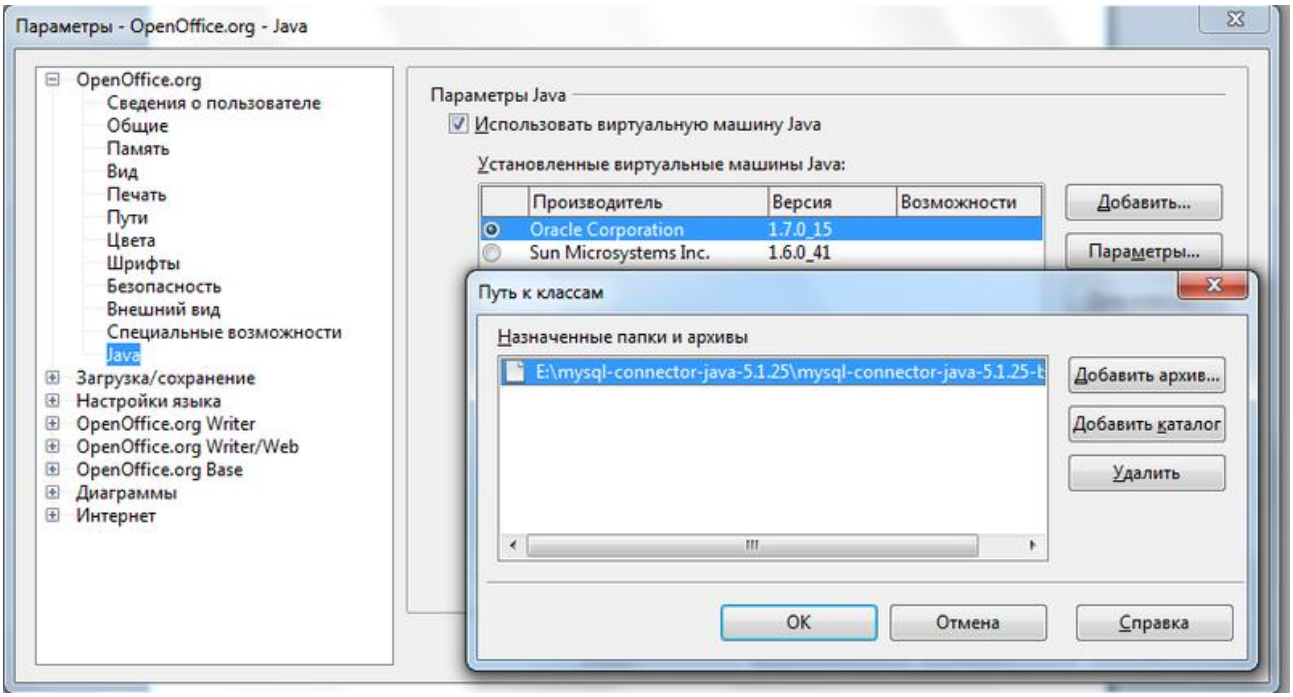

Рис. 4.7. Встановлення драйвера JDBC для MySQL

Далі перевіряємо роботу драйвера. Створюємо нову базу даних. Запускається майстер створення БД. Тут в якості джерела даних вибираємо JDBC. На другому кроці майстра необхідно виконати настройку з'єднання. Задаємо параметри налаштування з'єднання з базою MySQL через JDBC:

- Вказуємо Url джерела даних у форматі mysql://ім'я_сервера: 3306/ Імя_бази_данних_в_MySQL;
- Вказуємо клас драйвера JDBC: Для драйвера JDBC-MySQL це com.mysql.jdbc.Driver

Натискаємо кнопку "Перевірити клас". При натисканні кнопки відбувається перевірка завантаження JDBC – драйвера (рисунок 4.8).

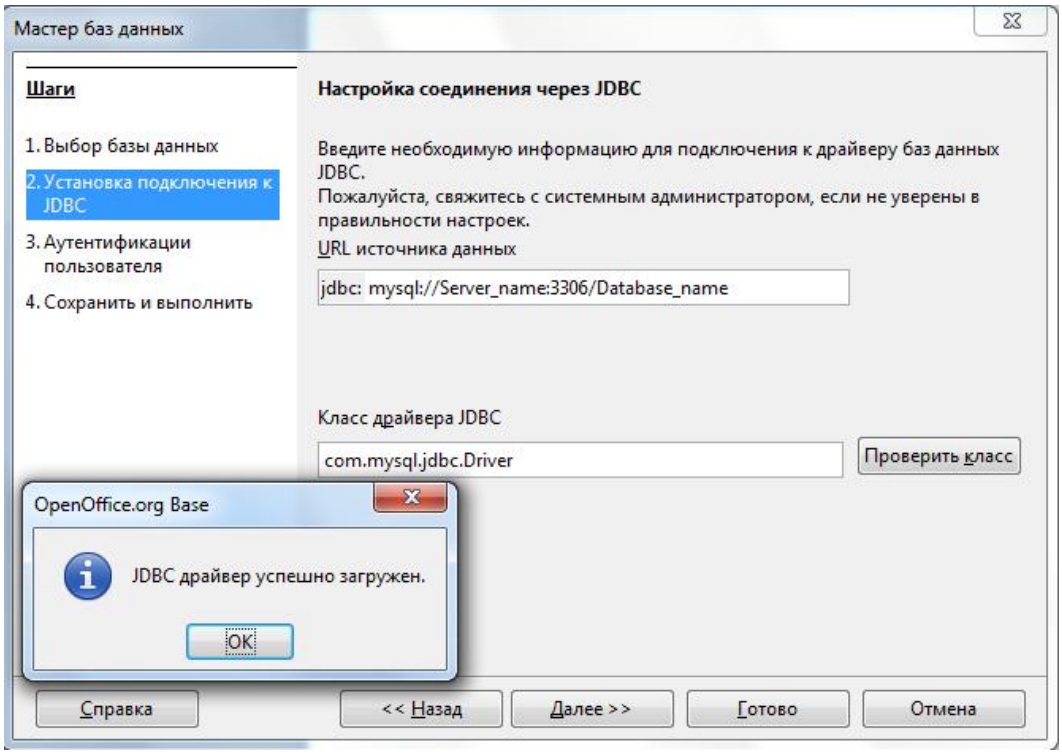

Рис. 4.8. Налаштування драйвера JDBC для MySQL

На наступному кроці пропонується ввести пароль та ім’я користувача з'єднання з MySQL для перевірки та встановлення з'єднання. Налаштування драйвера та проекту виконанено.

4.3. Інструкція користувача

Робота із системою обліку споживання води розпочинається із аутентифікації користувача. На рисунку 4.9 представлено форму аутентифікації користувача в системі.

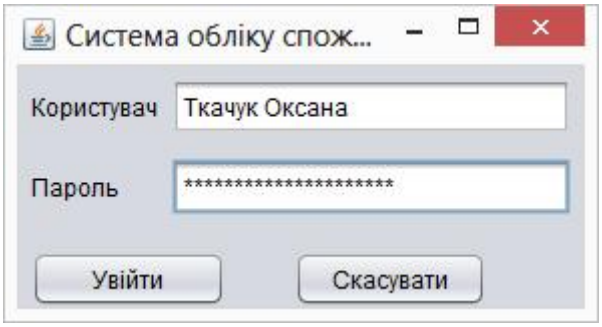

Рис. 4.9. Форма аутентифікації користувача

Нижче на рисунках (4.10 – 4.15) представлені основні екранні форми розробленої програми. Журнал оплат містить відомості про вже існуючі оплати фізичних осіб (рисунок 4.10) та юридичних осіб (рисунок 4.11).

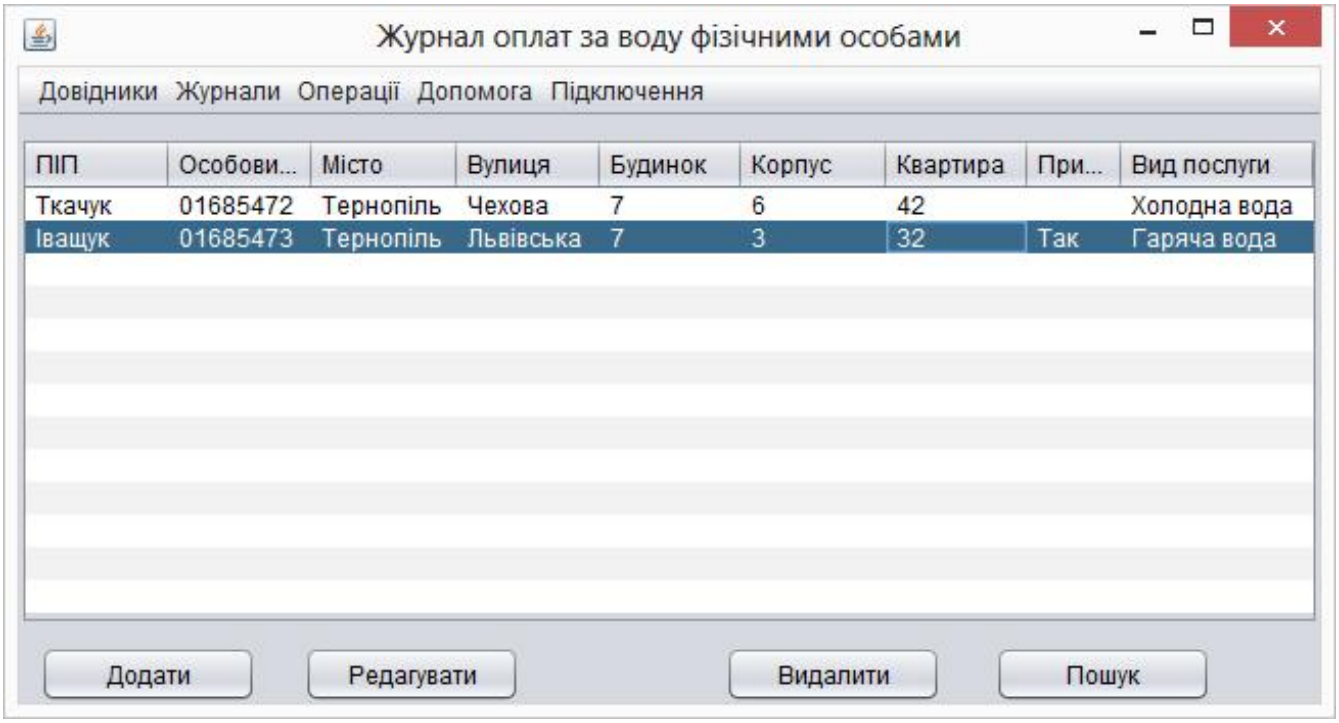

Рис. 4.10. Журнал оплат фізичних осіб

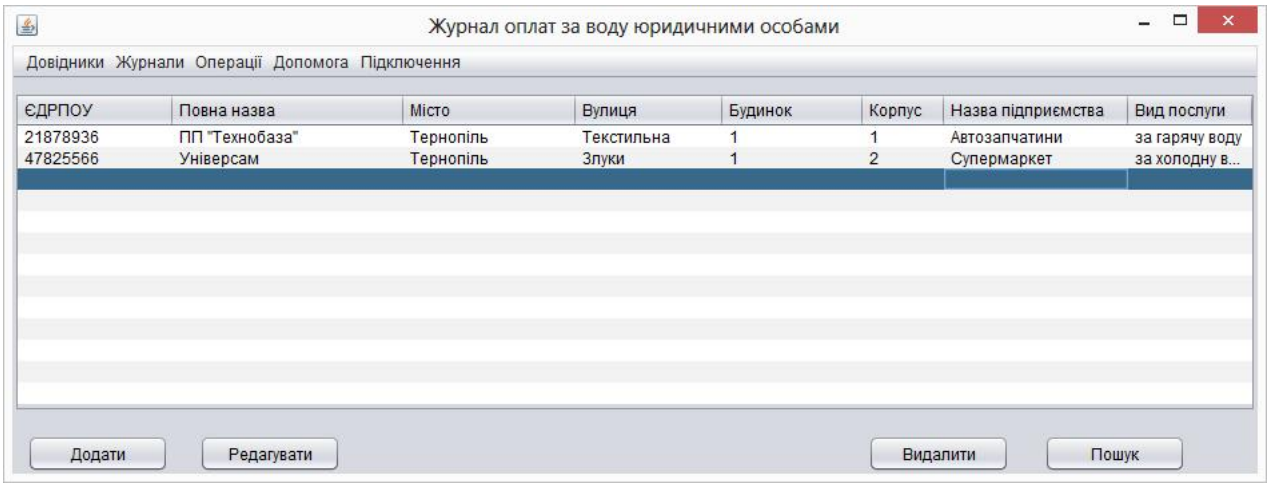

Рис. 4.11. Журнал оплат юридичних осіб

Журнали оплат містять відомості з карток абонентів (рисунок 4.12), які в свою чергу складають журнал фізичних осіб, та відомості з карток підприємств, які в свою чергу складають журнал підприємств.

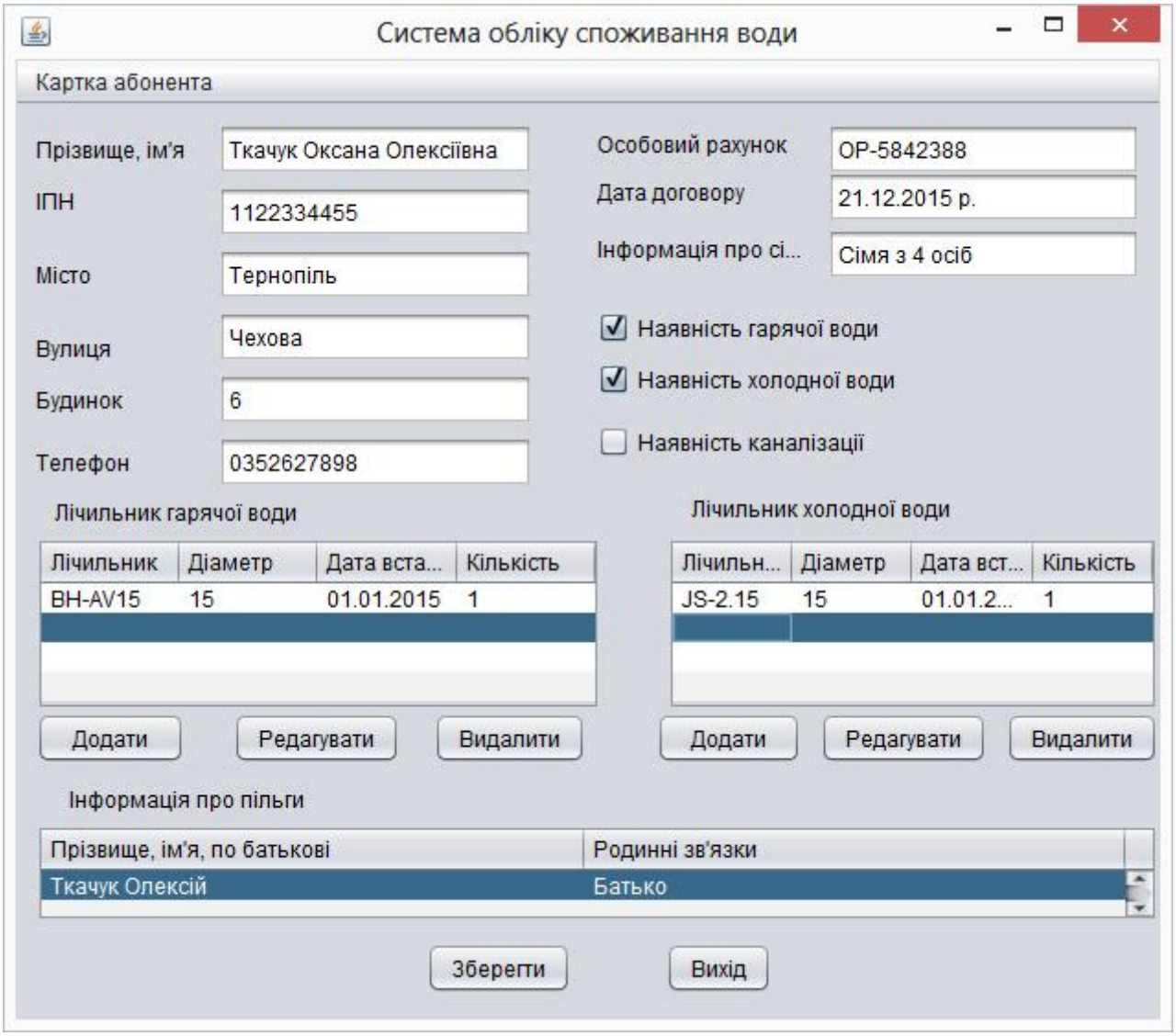

Рис. 4.12. Картка абонента

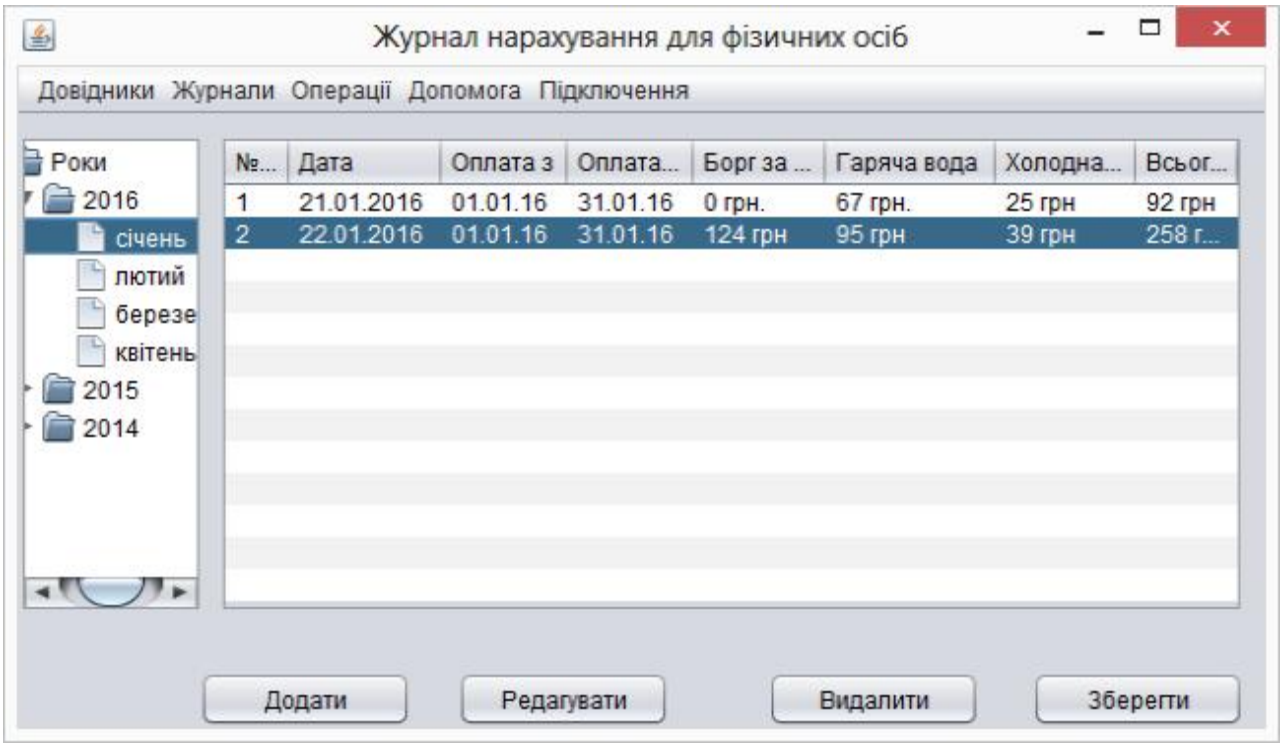

Рис. 4.13. Журнал нарахувань для фізичних осіб

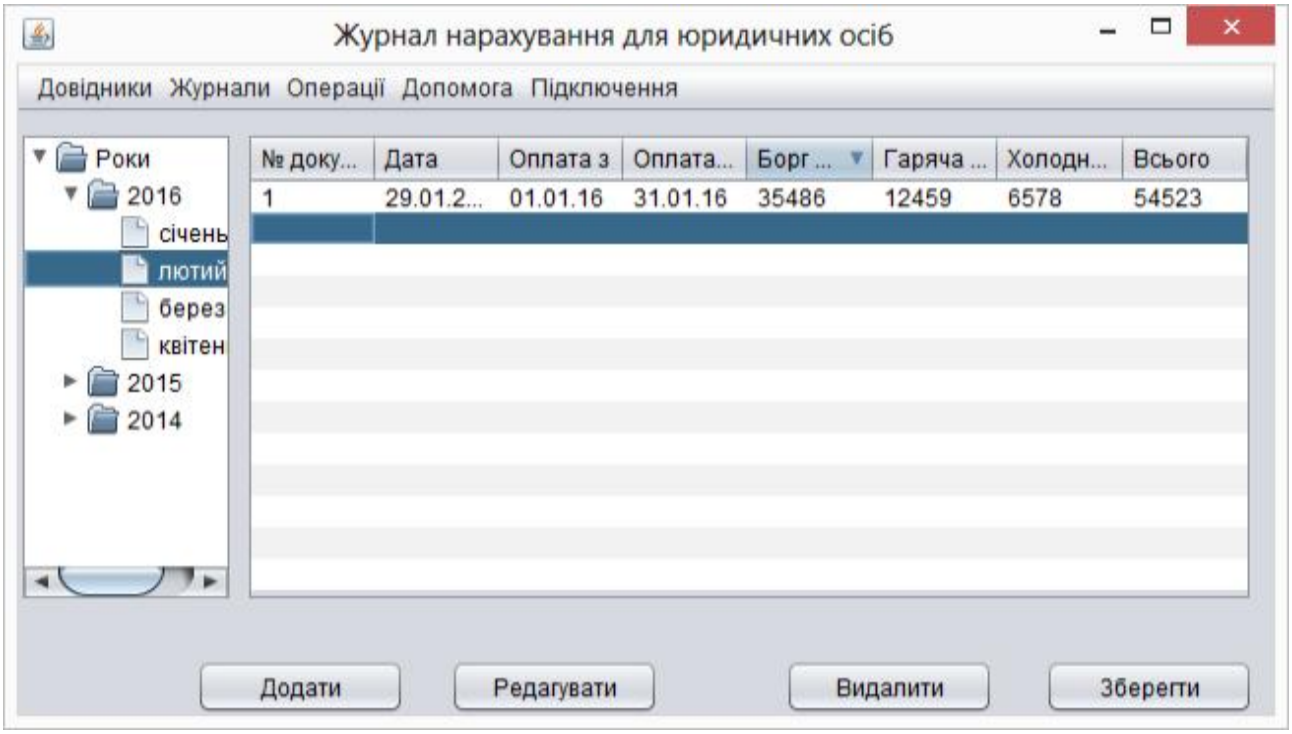

Рис. 4.14. Журнал нарахувань для фізичних осіб

Проведення розрахунків та видача результатів для населення представлена на рисунку 4.15

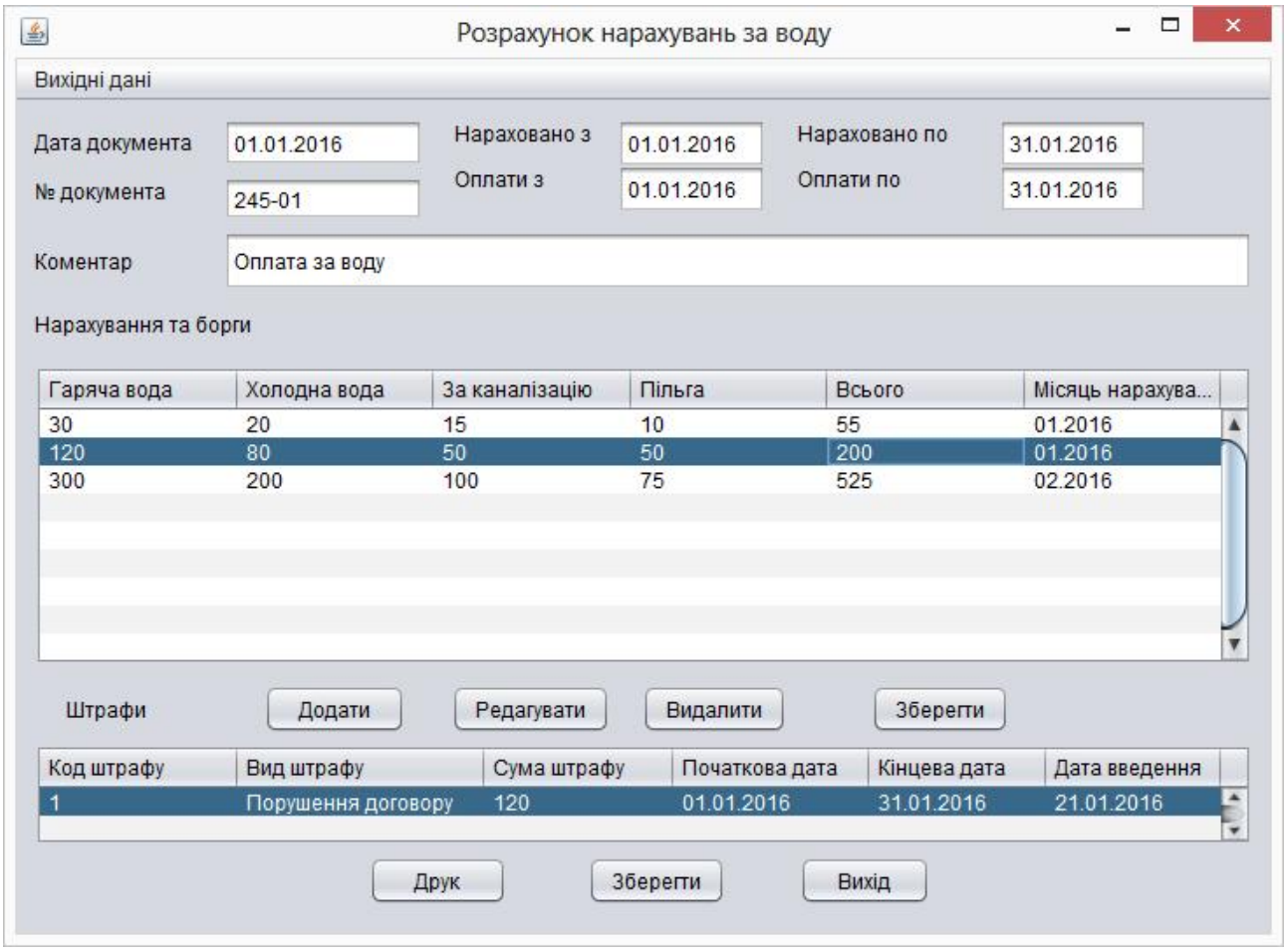

Рис. 4.15. Розрахунок нарахувань для фізичних осіб

Також в системі реалізована підсистема звітності, яка дозволяє отримати оперативну інформацію в розрізі заданих ознак, що значно пришвидшує швидкість прийняття рішень.

Висновки до розділу 4

Здійснено опис процедур тестування та їхніх результатів, описані тест-вимоги до програмного забезпечення, а також виявлені дефекти. Розкрито питання встановлення та налаштування програмного забезпечення на сервері, а також вказані вимоги, дотримання яких необхідно для користування системою, описана інструкція користувача для роботи із системою.

ВИСНОВКИ

При розрахунках зі споживачами послуг водопровідно-каналізаційного господарства використовується великі обсяги документації, звітів, які на сьогоднішній день не можливо обробити без використання автоматизованих систем. Також для коректної роботи необхідно звертатися до великої кількості баз даних та довідників, тому існуючі автоматизовані системи намагаються охопити великий обсяг робіт.

Програма, яка використовується в dиробничому управлінні водопровідно-каналізаційного господарства, потребує модернізації для підвищення ефективності та поліпшення функціонування всієї системи.

Висока вартість систем розрахунків зі споживачами, які представлені на ринку систем абонентського обліку, з одного боку, та неможливість редагування штатними програмістами програми для задоволення потреб підприємства та зміни шаблонів відомостей і звітів, з іншого, дають всі підстави для розробки власного програмного забезпечення.

В рамках виконання бакалаврської роботи реалізована програмна система обліку споживання води, яка на відміну від відомих рішень у цій галузі, реалізована за допомогою безкоштовних програмних технологій, і орієнтована чітко під вимоги замовника програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Агаджанов Г.К. Економіка водопровідно-каналізаційних підприємств. Навчальний посібник. – Х: Основа, 2000.
2. Буров Є. В. Комп’ютерні мережі: підручник / Євген Вікторович Буров. — Львів: «Магнолія 2006», 2010. — 262 с. ISBN 966-8340-69-8
3. Калянов Г.К. Case-технологии. Консалтинг при автоматизации бизнес-процессов (3 издание). – М.: Горячая линия – Телеком, 2002. – 320 с.
4. Ландэ Д. В., Снарский А. А., Безсуднов И. В. Интернетика: Навигация в сложных сетях: модели и алгоритмы. — М.:Либроком (Editorial URSS), 2009. — 264 с. ISBN=978-5-397-00497-8.
5. Комп’ютерні мережі: [навчальний посібник] / А. Г. Микитишин, М. М. Митник, П. Д. Стухляк, В. В. Пасічник. — Львів: «Магнолія 2006», 2013. — 256 с. ISBN 978-617-574-087-3
6. Инженерное программирование для проектирования программного обеспечения. -М.: Радио і связь, 1985, -512с.
7. Бойков.В., Савинков В.М. Проектирование баз данных информационных систем. М. Мир 1997
8. Бердтис А. Структуры данних. - М.: Статистика, 1974, - 408 с.
9. Горбань О.М., Бахрушин В.Є. Основі теорії систем та системного аналізу. - Запоріжжя, ГУ "ЗІДМУ", 2004, ISBN 966-8227-23-9
10. Майо Д. Самоучитель Microsoft Visual Studio 2010 = Microsoft Visual Studio 2010: A Beginner’s Guide (A Beginners Guide). — С.: «БХВ-Петербург», 2010. — С. 464. — ISBN 978-5-9775-0609-0
11. Алекс Макки Введение в .NET 4.0 и Visual Studio 2010 для профессионалов = Introducing .NET 4.0: with Visual Studio 2010. — М.: «Вильямс», 2010. — С. 416. — ISBN 978-5-8459-1639-6
12. Кен Хендерсон Професійне керівництво з SQL Server: структура та реалізація. — М.: Издательский дом «Вильямс», 2006. — С. 1056. ISBN 5-8459-0912-0

13. Чураков Михаил. Муравьиные алгоритмы [Електронний ресурс] / Михаил Чураков,Андрей Якушев. // Режим доступу: http://rain.ifmo.ru/cat/data/theory/unsorted/ant-algo-2006/article.pdf.
14. Бормашов Д. А. “Кластерный анализ текстов”: Дипломная работа [Електронний ресурс] /Д. А. Бормашов. Режим доступу: http://inf.tsu.ru/library/DiplomaWorks/CompScience/2006/bormashov/diplom.pdf.
15. Чубукова И.А. Data Mining БИНОМ. Лаборатория знаний, Интернет-университет информационных технологий - ИНТУИТ.ру, 2006.
16. Дюк В.А., Самойленко А.П. Data Mining: учебный курс. – СПб.: Питер, 2001.
17. Барсегян А. А., Куприянов М. С., Степаненко В. В., Холод И. И. Методы и модели анализа данных: OLAP и Data Mining. – СПб.: БХВ-Петербург, 2004. – 336 с.
18. Дивак М. П., Шпінталь М.Я., Козак О.Л., Струбицька І.П., Спільчук В.М., Піговський Ю.Р. Методичні рекомендації до виконання дипломної роботи освітньо кваліфікаційного рівня «бакалавр» клієнтам усіх форм навчання для напряму підготовки 6.050103 – «Програмна інженерія» // Тернопіль : ФОП Шпак П.П. - 2014. - 54 с.

ДОДАТОК А

ЛІСТИНГ ОСНОВНИХ МОДУЛІВ СИСТЕМИ

```
import java.sql.*;
import java.util.*;

public class TableAnalyser {

  Connection connection;
  String DBName;

  public TableAnalyser(Connection connection, String DBName, String DirectoryMeta,
String ColumnMeta, String ReadOnlyMeta) {
    this.connection = connection;
    this.DBName = DBName;
   }
   public Vector analyseTable(String TableName) {
    Vector fields = new Vector();
    HashMap mirror = new HashMap();
    try {
      DatabaseMetaData meta = connection.getMetaData();
      ResultSet result = meta.getColumns(DBName, null, TableName, null);

      while (result.next()) {
        int column_type = result.getInt("DATA_TYPE");
        if (column_type < 0)
          continue;
        String name = result.getString("COLUMN_NAME");
        FieldInfo field = new FieldInfo(name, column_type);

        field.setSize(result.getInt("COLUMN_SIZE"));
        fields.add(field);
        mirror.put(name, field);
      }
```

```
    result.close();

    result = meta.getPrimaryKeys(DBName, null, TableName);
    FieldInfo fi;
    while (result.next()) {
      fi = (FieldInfo)mirror.get(result.getString("COLUMN_NAME"));
      fi.setPrimary(true);
    }
    result.close();

    result = meta.getImportedKeys(DBName, null, TableName);
    while (result.next()) {
      fi = (FieldInfo)mirror.get(result.getString("FKCOLUMN_NAME"));
      fi.setForeign(true);
      fi.setForeignTable(result.getString("PKTABLE_NAME"));
      fi.setForeignField(result.getString("PKCOLUMN_NAME"));
            ResultSet fresult = meta.getColumns(DBName, null, fi.getForeignTable(), null);

      fresult.next();
      fresult.next();
      fi.setForeignFace(fresult.getString("COLUMN_NAME"));
      fresult.close();

    }
    result.close();
  }
  catch (SQLException ex) {
    ex.printStackTrace();
  }
  return fields;
}

public Connection getConnection() {return connection;}
```

```
        PreparedStatement generateInsert(String TableName, Vector structure) throws
SQLException {
          StringBuffer insert_string = new StringBuffer("INSERT INTO ");
          insert_string.append(TableName);
          insert_string.append(" (");

          int n = 0;
          FieldInfo info;
          for (Iterator i = structure.iterator(); i.hasNext(); ) {
            info = (FieldInfo)i.next();
            if (info.isPrimary()) {
              continue;
            }
            insert_string.append(info.getName());

            if (i.hasNext())
              insert_string.append(",  ");
            ++n;
          }

          insert_string.append(") VALUES (");

          for (int j = 0; j < n - 1; ++j)
            insert_string.append("?,  ");

          insert_string.append("?");
          insert_string.append(")");


          return connection.prepareStatement(insert_string.toString());
        }

        PreparedStatement generateUpdate(String TableName, Vector structure) throws
            SQLException {
          String primary_name = "id";
```

```
    StringBuffer put_string = new StringBuffer("UPDATE ");
    put_string.append(TableName);
    put_string.append(" SET ");

    FieldInfo info;
    for (Iterator i = structure.iterator(); i.hasNext(); ) {
      info = (FieldInfo) i.next();
      if (info.isPrimary()) {
        primary_name = info.getName();
        continue;
      }

      put_string.append(info.getName());
      if (i.hasNext()) {
        put_string.append(" = ?, ");
      }
      else
        put_string.append(" = ? ");
    }

    put_string.append(" WHERE ");
    put_string.append(primary_name);
    put_string.append(" = ?");

    return connection.prepareStatement(put_string.toString());
  }

}
public class FieldInfo {
  private String field_name;
  private int field_type;
  private boolean is_forein = false;
  private boolean is_primary = false;
  private String foreign_table;
```

```
  private String foreign_field;
  private String foreign_subst;
  private int field_size;

  public FieldInfo() {}
  public FieldInfo(String name, int type) {
    field_name = name;
    field_type = type;
  }
  public String getName() {return field_name;}
  public int getType() {return field_type;}
  public boolean isForeign() {return is_forein;}
  public boolean isPrimary() {return is_primary;}
  public String getForeignTable() {return foreign_table;}
  public String getForeignField() {return foreign_field;}
  public String getForeignFace() {return foreign_subst;}
  public int getSize() {return field_size;}

  public void setName(String v) {field_name = v;}
  public void setType(int v) {field_type = v;}
  public void setForeign(boolean v) {is_forein = v;}
  public void setPrimary(boolean v) {is_primary = v;}
  public void setForeignTable(String v) {foreign_table = v;}
  public void setForeignField(String v) {foreign_field = v;}

  public void setForeignFace(String v) {foreign_subst = v;}
  public void setSize(int v) {field_size = v;}
}
```

ДОДАТОК Б

DDL БАЗИ ДАНИХ

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `Streets`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Streets` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `Streets` (
  `StreetID` INT NOT NULL ,
  `StreetCode` VARCHAR(255) NULL ,
  `StreetName` VARCHAR(255) NULL ,
  PRIMARY KEY (`StreetID`) )
ENGINE = InnoDB;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `Houses`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Houses` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `Houses` (
  `HouseID` INT NOT NULL ,
  `CityID` INT NULL ,
  `HouseNumber` VARCHAR(10) NULL ,
  `HouseFraction` VARCHAR(10) NULL ,
  `HouseCorps` VARCHAR(10) NULL ,
  `LetterID` VARCHAR(45) NULL ,
  `StreetId` INT NULL ,
  `fPrivateSector` INT NULL ,
  `MirkoReekedID` TINYINT(1) NULL ,
  `Streets_StreetID` INT NOT NULL ,
  PRIMARY KEY (`HouseID`) )
ENGINE = InnoDB;

SHOW WARNINGS;
CREATE INDEX `fk_Houses_Streets1` ON `Houses` (`Streets_StreetID` ASC) ;

SHOW WARNINGS;
```

```
-- -----------------------------------------------------
-- Table `Benefits`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Benefits` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `Benefits` (
  `BenefitID` INT NOT NULL ,
  `BenefitName` VARCHAR(255) NULL ,
  `PercentConsessior` DECIMAL(10) NULL ,
  `dInserted` DATE NULL ,
  `BenefitShortName` VARCHAR(255) NULL ,
  `BenefitCode` VARCHAR(255) NULL ,
  PRIMARY KEY (`BenefitID`) )
ENGINE = InnoDB;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `Subscribers`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Subscribers` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `Subscribers` (
  `SubscriberID` INT NOT NULL ,
  `BenefitID` INT NULL ,
  `fSeverage` DECIMAL(10) NULL ,
  `fHotWater` DECIMAL(10) NULL ,
  `fCooltWater` DECIMAL(10) NULL ,
  `HouseID` INT NULL ,
  `ApartmentNum` INT NULL ,
  `AmountLodger` INT NULL ,
  `PersonalAccount` VARCHAR(255) NULL ,
  `SubscriberFio` VARCHAR(255) NULL ,
  `SubscriberTel` VARCHAR(50) NULL ,
  `dAgreement` DATE NULL ,
  `AmountLand` VARCHAR(45) NULL ,
  `Houses_HouseID` INT NOT NULL ,
  `Benefits_BenefitID` INT NOT NULL ,
  PRIMARY KEY (`SubscriberID`) )
ENGINE = InnoDB;

SHOW WARNINGS;
CREATE INDEX `fk_Subscribers_Houses1` ON `Subscribers` (`Houses_HouseID` ASC) ;

SHOW WARNINGS;
CREATE INDEX `fk_Subscribers_Benefits1` ON `Subscribers` (`Benefits_BenefitID`
ASC) ;

SHOW WARNINGS;
```

```
-- -----------------------------------------------------
-- Table `SubscribersFines`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SubscribersFines` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `SubscribersFines` (
  `SubscribersFineID` INT NOT NULL ,
  `SubscriberID` INT NULL ,
  `FineID` INT NULL ,
  `dPeriodStart` DATE NULL ,
  `dPeriodEnd` DATE NULL ,
  `dInserted` DATE NULL ,
  `Subscribers_SubscriberID` INT NOT NULL ,
  PRIMARY KEY (`SubscribersFineID`) )
ENGINE = InnoDB;

SHOW WARNINGS;
CREATE INDEX `fk_SubscribersFines_Subscribers1` ON `SubscribersFines` (`Subscribers_SubscriberID` ASC) ;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `Taxs`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `Taxs` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `Taxs` (
  `TaxID` INT NOT NULL ,
  `NumDocument` VARCHAR(25) NULL ,
  `dDocument` DATE NULL ,
  `dPeriodStart` DATE NULL ,
  `dPeriodENd` DATE NULL ,
  `dPaymentStart` DATE NULL ,
  `dPaymentEnd` DATE NULL ,
  `SumDebt` DECIMAL(10) NULL ,
  `SumTotal` DECIMAL(10) NULL ,
  `SumTotalCoolWater` DECIMAL(10) NULL ,
  `SumTotalHotWater` DECIMAL(10) NULL ,
  `SumTotalSprined` DECIMAL(10) NULL ,
  `Remark` VARCHAR(255) NULL ,
  `fConduct` TINYINT(1) NULL ,
  PRIMARY KEY (`TaxID`) )
ENGINE = InnoDB;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `TaxDetails`
-- -----------------------------------------------------
```

```
DROP TABLE IF EXISTS `TaxDetails` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `TaxDetails` (
  `TaxDetailID` INT NOT NULL ,
  `TaxID` INT NULL ,
  `SubscriberID` INT NULL ,
  `CoolWaterCube` DECIMAL(10) NULL ,
  `HotWaterCube` DECIMAL(10) NULL ,
  `SeverageCube` DECIMAL(10) NULL ,
  `SumTotalCoolWater` DECIMAL(10) NULL ,
  `SumTotalHotWater` DECIMAL(10) NULL ,
  `SumTotalHotSeverage` DECIMAL(10) NULL ,
  `SumTotalHotSprined` DECIMAL(10) NULL ,
  `SeveragePrivilege` DECIMAL(10) NULL ,
  `SumDebitCoolWater` DECIMAL(10) NULL ,
  `SumDebitHotWater` DECIMAL(10) NULL ,
  `SumDebitSeverager` DECIMAL(10) NULL ,
  `SumDebitSprined` DECIMAL(10) NULL ,
  `SumTotalDebit` DECIMAL(10) NULL ,
  `TaxDate` DATE NULL ,
  `SumTotal` DECIMAL(10) NULL ,
  `SumClean` DECIMAL(10) NULL ,
  `SumTotalPrivilege` DECIMAL(10) NULL ,
  `Taxs_TaxID` INT NOT NULL ,
  `Subscribers_SubscriberID` INT NOT NULL ,
  PRIMARY KEY (`TaxDetailID`) )
ENGINE = InnoDB;

SHOW WARNINGS;
CREATE INDEX `fk_TaxDetails_Taxs` ON `TaxDetails` (`Taxs_TaxID` ASC) ;

SHOW WARNINGS;
CREATE INDEX `fk_TaxDetails_Subscribers1` ON `TaxDetails` (`Subscribers_SubscriberID` ASC) ;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `BenefitPerson`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `BenefitPerson` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `BenefitPerson` (
  `BenefitPersonID` INT NOT NULL ,
  `SubscriberID` INT NULL ,
  `RelationDegreeID` INT NULL ,
  `BenefitFIO` VARCHAR(255) NULL ,
  `Subscribers_SubscriberID` INT NOT NULL ,
  PRIMARY KEY (`BenefitPersonID`) )
ENGINE = InnoDB;
```

```
SHOW WARNINGS;
CREATE INDEX `fk_BenefitPerson_Subscribers1` ON `BenefitPerson` (`Subscribers_SubscriberID` ASC) ;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `PayTransacts`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `PayTransacts` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `PayTransacts` (
  `PayTransactID` INT NOT NULL ,
  `SubscriberID` VARCHAR(45) NULL ,
  `PaymentTypeID` VARCHAR(45) NULL ,
  `CountDisplayStart` DECIMAL(10) NULL ,
  `CountDisplayEnd` DECIMAL(10) NULL ,
  `dPeriodStart` DATE NULL ,
  `dPeriodEnd` DATE NULL ,
  `PayDate` DATE NULL ,
  `SumDate` DECIMAL(10) NULL ,
  `Subscribers_SubscriberID` INT NOT NULL ,
  PRIMARY KEY (`PayTransactID`) )
ENGINE = InnoDB;

SHOW WARNINGS;
CREATE INDEX `fk_PayTransacts_Subscribers1` ON `PayTransacts` (`Subscribers_SubscriberID` ASC) ;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `SrMeterReadingCWs`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SrMeterReadingCWs` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `SrMeterReadingCWs` (
  `SrMeterReadingCWID` INT NOT NULL ,
  `SubscriberID` INT NULL ,
  `CoolCounterStart` DECIMAL(10) NULL ,
  `CoolCounterEnd` DECIMAL(10) NULL ,
  `dPeriodStart` DATE NULL ,
  `dPeriodEnd` DATE NULL ,
  `CounterSumTotal` DECIMAL(10) NULL ,
  `Subscribers_SubscriberID` INT NOT NULL ,
  PRIMARY KEY (`SrMeterReadingCWID`) )
ENGINE = InnoDB;

SHOW WARNINGS;
```

```
CREATE INDEX `fk_SrMeterReadingCWs_Subscribers1` ON `SrMeterReadingCWs` (`Subscribers_SubscriberID` ASC) ;

SHOW WARNINGS;

-- -----------------------------------------------------
-- Table `SrMeterReadingHWs`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `SrMeterReadingHWs` ;

SHOW WARNINGS;
CREATE  TABLE IF NOT EXISTS `SrMeterReadingHWs` (
  `SrMeterReadingHWID` INT NOT NULL ,
  `SubscriberID` INT NULL ,
  `CoolCounterStart` DECIMAL(10) NULL ,
  `CoolCounterEnd` DECIMAL(10) NULL ,
  `dPeriodStart` DATE NULL ,
  `dPeriodEnd` DATE NULL ,
  `CounterSumTotal` DECIMAL(10) NULL ,
  `Subscribers_SubscriberID` INT NOT NULL ,
  PRIMARY KEY (`SrMeterReadingHWID`) )
ENGINE = InnoDB;

SHOW WARNINGS;
CREATE INDEX `fk_SrMeterReadingHWs_Subscribers1` ON `SrMeterReadingHWs` (`Subscribers_SubscriberID` ASC) ;

SHOW WARNINGS;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```