Київський національний торговельно-економічний університет
Кафедра інформаційних технологій

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ
на тему:

« Створення програмного забезпечення для розподілу навчального навантаження кафедри»

(за матеріалами Кафедри інформаційних технологій, КНТЕУ)

Студента 2 курсу, 10м групи,
факультету обліку, аудиту та інформаційних систем,
денної форми навчання
спеціальності
122 «Комп'ютерні науки» _____ Колот Артем Віталійович

Науковий керівник
науковий ступінь
вчене звання _____ Нагірна А.М.
канд. фіз.-мат. наук, доцент

Гарант освітньої програми
науковий ступінь
вчене звання _____ Краскевич В. Є.
доктор технічних наук, професор

Київ 2018

Зміст

ВСТУП

Вищий учбовий заклад є складною активною соціально-економічною системою. Управління такою системою представляє собою трудомісткий процес, потребуючий великих витрат ресурсів не тільки енергетичних, матеріальних, але і людських. Одна з основних завдань управління навчальним закладом полягає в оптимізації освітнього процесу за показниками, що характеризує цей процес.

Ефективне і якісне управління кафедрою засноване на застосуванні інформаційних технологій є одним з основних умов для її успішного розвитку, а також випуску затребуваних ринком праці фахівців. Що в свою чергу підвищує конкурентоспроможність кафедри на ринку освітніх послуг.

Проблемою автоматизації управління своїми підрозділами займаються багато ВНЗ України. На жаль, незважаючи на позитивні зрушення, що намітилися в останні роки, розподіл і облік виконання навчального навантаження до сих пір виконується вручну.

Прийняття керуючого рішення при організації роботи складної системи пов'язано з обробкою великого обсягу інформації, що вимагає нетривіального аналізу ситуації, що склалася (із залученням методів, розроблених в самих різних областях знань) і повинно бути своєчасним (тоді як часто на прийняття рішення відводиться відносно малий проміжок часу). Прогрес суспільства з кожним роком загострює цю проблему. Саме тому в сучасному системному аналізі основні зусилля спрямовуються на створення математичних моделей управління складними системами, які дозволили б розробити відповідні інформаційно-комп'ютерні технології з мінімальною участю людини в процесі управління.

Ефективне управління навчальним процесом у ВНЗ сьогодні потребує переходу на якісно нові технології роботи з даними, що відносяться до навчального процесу, засновані на використанні комп'ютерних мереж і баз даних.

Навчальне навантаження - це основа робочого часу викладача, встановлювана керівником (завідувачем кафедрою), виходячи з професійної компетентності викладача, кількості годин за навчальним планом та навчальних програм, забезпеченості кадрами, та інших особливостей установи.

Специфіка об'єкта управління, яким є освітня діяльність, і слабкий розвиток інформаційних систем для освітньої галузі роблять актуальним завдання розробки автоматизованих систем інформаційної підтримки управління ресурсами процесів життєвого циклу освітньої діяльності ВНЗ та його основних виробничих підрозділів - кафедр.

Метою дипломної роботи є підвищення ефективності роботи підрозділів у вищих учбових закладах та зменшення витрати тимчасових і людських ресурсів на розподіл навантаження на кафедрі.

Об'єктом дослідження є процес автоматизації розподілу педагогічного навантаження.

Предметом дослідження є розподіл педагогічного навантаження у вищих учбових закладах.

На сучасному рівні розвитку інформаційних технологій до сих пір недостатньо розроблені методи розподілу навчального навантаження, що враховують безліч різних чинників.

На розподіл педагогічного навантаження впливають наступні фактори:

- частка участі кожного викладача в роботі кафедри (штатна одиниця);
- кількість викладачів, які ведуть заняття в навчальній групі;
- кількість студентів у навчальній групі;
- наявність лекційних годин заданого потоку у конкретного викладача;
- кількість навчальних груп заданого потоку у конкретного викладача.

При автоматизації процесу розподілу та обліку виконання навчального навантаження повинні бути враховані наступні вимоги:

- Скорочення часу на введення інформації з первинних документів: з навчального відділу повинен надходити електронний варіант документа «Розрахунок навантаження на кафедру».
- Автоматизація процесу розподілу дисциплін між викладачами відповідно до посади,
- переліку дисциплін, що викладаються, діючими нормами.
- Можливість редагування педагогічного навантаження вручну відповідно до уподобань і побажань викладачів.
- Можливість автоматичного контролю правильності розподілу навантаження.
- Автоматичне формування вихідних документів і звітів.

1 ОГЛЯД ТА АНАЛІЗ ОБАСТІ РОЗПОДІЛУ ПЕДАГОГІЧНОГО НАВАНТАЖЕННЯ

1.1 Встановлення та розподіл педагогічного навантаження. Основні поняття та визначення

Встановлення педагогічного навантаження педагогічним працівникам і оплата їх праці проводиться у відповідності до 25 Закону України «Про загальну середню освіту» від 13 травня 1999 року № 651-XIV (з наступними змінами) (далі – Закон № 651-XIV) та Інструкції про порядок обчислення заробітної плати працівників освіти, затвердженої наказом Міносвіти України від 15.04.93 № 102 (з наступними змінами), зареєстрованим у Міністерстві юстиції України 27 травня 1993 р. за № 56 (далі – Інструкція № 102).

Стаття 25 Закону № 651-XIV визначає:

- поняття педагогічного навантаження вчителя загальноосвітнього навчального закладу незалежно від підпорядкування, типу і форми власності;
- обсяги навчального та педагогічного навантаження;
- інші види педагогічної діяльності;
- розміри доплат за інші види педагогічної діяльності;
- обсяг педагогічне навантаження вихователів загальноосвітнього навчального закладу та загальноосвітньої спеціальної школи (школи-інтернату).

Так, педагогічне навантаження викладача вищого навчального закладу, це час, призначений для здійснення навчально-виховного процесу, та включає як навчальні години, так і інші види педагогічної діяльності. Обсяг навчального навантаження та іншої педагогічної діяльності, які може виконувати педагогічний працівник за основним місцем роботи, граничними розмірами не обмежується.

Перерозподіл педагогічного навантаження протягом навчального року допускається у разі зміни кількості годин з окремих предметів, що передбачається робочим навчальним планом, або за письмовою згодою педагогічного працівника з додержанням законодавства України про працю. Педагогічне навантаження викладача вищого навчального закладу незалежно від підпорядкування, типу і форми власності обсягом менше тарифної ставки, передбаченої частиною першою статті 25 Закону № 652-XIV, встановлюється тільки за його згодою.

Навчальне навантаження між викладачами та іншими педагогічними працівниками розподіляється керівником установи за погодженням з профспілковим комітетом залежно від кількості годин, передбачених навчальними планами, наявності відповідних педагогічних кадрів та інших конкретних умов, що склались у закладі (з дотриманням Кодексу законів про працю України). Обсяг навчального навантаження педагогічним працівникам шкіл, шкіл-інтернатів (крім учителів вечірніх (змінних) середніх загальноосвітніх шкіл (класів) з очно-заочною формою навчання, заочних шкіл, а також учителів, що навчають дітей, які перебувають на тривалому лікуванні у лікарнях) і викладачам педучилищ визначається один раз на рік окремо по півріччях. Навчальне навантаження вчителів вечірніх (змінних) середніх загальноосвітніх шкіл (класів) з очно-заочною формою навчання, заочних шкіл, а також вчителів, що навчають дітей, які перебувають на тривалому лікуванні у лікарнях, визначається два рази на рік до початку першого та другого півріччя. Навчальне навантаження викладачам вищих навчальних закладів I - II рівнів акредитації та професійно-технічних навчальних закладів визначається один раз на рік до початку навчального року.

Педагогічне навантаження вчителя включає 18 навчальних годин протягом навчального тижня, що становлять тарифну ставку. За інші види педагогічної діяльності встановлюються доплати в такому розмірі:

- класне керівництво20–25 відсотків посадового окладу
- перевірка зошитів 10–20 відсотків посадового окладу
- завідування

- майстернями 15–20 відсотків
- навчальними кабінетами 10–15 відсотків посадового окладу
- навчально-дослідними ділянками 10–15 відсотків посадового окладу

Педагогічне навантаження вихователя загальноосвітнього навчального закладу становить 30 годин, вихователя загальноосвітньої спеціальної школи (школи-інтернату) – 25 годин на тиждень, що становить тарифну ставку. Оплата праці педагогічних працівників установ і закладів освіти провадиться виходячи із встановлених ставок заробітної плати (посадових окладів) з врахуванням підвищень, фактичного обсягу педагогічної роботи, доплат та надбавок.

Відповідно до підпункту а) пункту 64 Інструкції № 102 ставки заробітної плати (посадові оклади) вчителів, викладачів, вихователів та інших педагогічних працівників виплачуються (норма годин на ставку встановлена, виходячи з 6-денного режиму роботи):

1) за 3 години педагогічної (викладацької) роботи на день (18 годин на тиждень): вчителям I - XI (XII) класів шкіл та шкіл-інтернатів усіх типів і найменувань; вчителям трудового навчання II - III класів спеціальних шкіл для дітей та підлітків,які потребують особливих умов виховання;учителям-логопедам, учителям-дефектологам,практичним психологам спеціальних загальноосвітніх шкіл (шкіл-інтернатів);викладачам III - V класів шкіл естетичного виховання (загальної музичної, художньої, хореографічної освіти) з п'ятирічним строком навчання;викладачам V - VII класів шкіл естетичного виховання (дитячих музичних, художніх, хореографічних шкіл, шкіл мистецтв) з семирічним строком навчання;викладачам спеціальних дисциплін I - XI класів середніх музичних і художніх шкіл та шкіл-інтернатів;викладачам I - IV класів дитячих художніх шкіл і шкіл загальної художньої освіти з чотирирічним строком навчання;викладачам і старшим викладачам педагогічних училищ;учителям-дефектологам і логопедам установ охорони здоров'я та соціального забезпечення (крім будинків дитини);керівникам гуртків, секцій, студій та інших форм гурткової роботи, в тому числі

організованих у закладах освіти на принципах самоокупності;учителям груп для занять з іноземної мови в початкових класах загальноосвітніх шкіл і в дошкільних закладах, що працюють на принципах самоокупності;викладачам постійно діючих курсів по вивченню мов, стенографії, машинопису з терміном навчання один рік і більше;вчителям і викладачам училищ фізичної культури.

2) за 3 години викладацької роботи в середньому в день (720 годин на рік):викладачам та старшим викладачам вищих навчальних закладів I - II рівнів акредитації та професійно-технічних навчальних постійно діючих курсів, курсів по підготовці до вступу в вищі навчальні заклади I - II рівнів акредитації.

3) за 20 годин викладацької роботи на тиждень:завідуючим логопедичними пунктами, логопедам будинків дитини, вчителям-логопедам і вчителям-дефектологам у дошкільних навчальних закладах.

4) за 4 години педагогічної (викладацької) роботи в день (24 години на тиждень):викладачам I - II класів шкіл естетичного виховання (загальної музичної, художньої, хореографічної освіти) з п'ятирічним строком навчання;викладачам I - IV класів шкіл естетичного виховання (дитячих музичних, художніх, хореографічних шкіл, шкіл мистецтв) з семирічним строком навчання;музичним керівникам;концертмейстерам, акомпаніаторам і культорганізаторам.

5) за 4 години педагогічної (викладацької) роботи на день (1016 годин на рік):викладачам постійно діючих курсів по вивченню мов, стенографії, машинопису з строком навчання менше одного року.

6) за 25 годин виховної роботи на тиждень:вихователям спеціальних навчально-виховних закладів (груп) для дітей і підлітків з вадами у фізичному чи розумовому розвитку.

7) за 5 годин педагогічної роботи в день (30 годин на тиждень):вихователям груп загального типу дошкільних навчальних закладів, вихователям, старшим вихователям шкіл-інтернатів, дитячих будинків, крім шкіл-інтернатів з поглибленим вивченням російської мови та посиленою військово-фізкультурною підготовкою, інтернатів, шкіл (груп) продовженого

дня, установ охорони здоров'я, приймальників-розподільників для неповнолітніх, виховно-трудових колоній, спеціальних шкіл та спеціальних профтехучилищ для дітей і підлітків, які потребують особливих умов виховання, кімнат школяра при клубах, палацах культури, житлово-експлуатаційних організаціях, училищ фізичної культури, інструкторам з фізкультури, інструкторам слухових кабінетів.

8) за 6 годин педагогічної роботи в день (36 годин на тиждень):вихователям-методистам дитячих дошкільних закладів, дитячих майданчиків, дитячих парків;

9) за 7 годин роботи в день (41 година на тиждень):майстрам виробничого навчання, в тому числі навчання водінню, вихователям професійно-технічних навчальних закладів.

Ставки заробітної плати педагогічних працівників, перелічених у підпунктах 1) – 9), встановлюються виходячи із затрат робочого часу в астрономічних годинах (60 хвилин). Короткі перерви, передбачені між уроками, заняттями, лекціями є робочим часом педагогічного працівника. Тривалість уроків і перерв встановлюється відповідно до положення про навчальний заклад. У навчальних планах шкіл усіх типів і найменувань, педучилищ, професійно-технічних навчальних закладів час на навчальні дисципліни визначається в академічних годинах (45 хвилин), крім перших класів шкіл, де навчаються діти шестирічного віку (35 хвилин). Скорочення тривалості уроку не є підставою для зменшення норми часу, що передбачається на вивчення предмета і не є підставою для зменшення норми часу для оплати в розрізі чверті, півріччя, року. Збільшення кількості уроків в зв'язку із зменшенням її тривалості не означає збільшення навчального навантаження, встановленого при тарифікації.

За години педагогічної роботи, виконаної понад встановлену норму, провадиться додаткова оплата відповідно до отримуваної ставки заробітної плати в одинарному розмірі. У тих випадках, коли робота понад встановлену норму виконується вихователями, помічниками вихователів, санітарками-нянями дошкільних закладів з причини невиходу на роботу змінника, або у випадках, коли батьки несвоєчасно забирають дітей з дошкільного закладу і

робота виконується за межами робочого часу, встановленого графіками роботи, оплата праці провадиться відповідно до статті 106 Кодексу законів про працю України, як за надурочну. Цей порядок застосовується у випадках, коли робота понад встановлену норму робочого часу виконувалась без перерви після закінчення основного робочого часу.

Адміністрація зобов'язана вжити заходів для заміни відсутнього працівника. Надурочна робота допускається, як виняток, з дозволу профспілкового комітету. Надурочна робота не повинна перевищувати для кожного працівника 120 годин на рік. При погодинній оплаті праці надурочна робота оплачується за перші дві години у полуторному розмірі, а наступні години – у подвійному розмірі. Оплата надурочних робіт здійснюється в межах фонду заробітної плати (фонду оплати праці) закладу, установи.Якщо для працівника установи чи закладу освіти законодавством не передбачено скороченої тривалості робочого дня, то він становить 7 годин при шестиденному робочому тижні (41 годину на тиждень).

1.2 Аналіз підходів до автоматизації розподілу навантаження

На даний момент на кафедрі існує система розподілу навантаження. Ця система - результат спроб реалізації інформаційної системи. Реалізована в програмному забезпеченні Microsoft Excel, який заснований на застосуванні макросів. Розробка програмної документації та програмного продукту повинна проводитися згідно з ГОСТ 19.701-90 , ГОСТ 2.304-88.

Система дозволяє:

- додавати нових викладачів до списку
- при розподілі навантаження бачити підсумкові дані в динамічному режимі, такі як сумарне значення навантаження в годинах для всіх видів занять, сумарне значення ставок викладачів (це можливо за рахунок зв'язування файлів)
- роздрукувати стандартні звіти, а саме Звіт та Навчальні доручення для кожного викладача
- вихідні документи зберігаються кожен в окремому документі.

Друга інформаційна система включає в себе дві підсистеми:

Перша з підсистем, представлених раніше, а саме «Розрахунок навчального навантаження викладацького складу і розподіл по кафедрам ВНЗ», припускає наявність таких функцій:

- проводиться додатковий контроль по семестрах на число курсових робіт і курсових проектів, число заліків та іспитів, на суму аудиторних занять на тиждень тощо;
- система оперативно повідомляє про всі відхилення від нормативних меж, встановлених Держстандартом, а також дозволяє роздрукувати протокол розбіжностей;
- отримана база графіків навчального процесу використовується для розрахунку штатів ППС університету та кафедри;
- при розрахунку штатів ППС враховуються аудиторні заняття, керівництво практикою, аспірантурою, докторантурою, участь у державних атестаційних комісіях і т.д.;
- програма оперативно виконує перерахунок навантаження викладача за будь-яких, внесених в інтерактивному режимі, змінах його навантаження;
- на екран виводяться попереджувальні повідомлення про перевищення навантаження викладача вище ставки, здійснює контроль за загальної навчальної навантаженням кафедри;
- при формуванні навчальних доручень програма групує навчальне навантаження викладача по семестрах;
- розподіл навантаження виробляється на основі списку предметів і списку викладачів, визначеного користувачем;

побудова попередньої опції дозволяє розподілити викладання одного предмета між кількома викладачами, передбачаючи всі можливі варіанти: проведення лекцій, практичних занять і лабораторних робіт (у тому числі ведення лабораторних робіт у різних підгрупах однієї групи) різними викладачами.

Друга з представлених підсистем – «Розподіл навчального навантаження викладачам кафедри», пропонує наступні функції:

- введення і зміна списків викладачів, навчального плану, а також довідників з переліками дисциплін і спеціальностей;
- зберігання інформації про загальний навчальний план кафедри та індивідуальної навантаженні кожного викладача;
- вибір і автоматичне занесення даних із загального в індивідуальний план викладача з одночасним перерахунком його підсумкового навантаження. Вибір здійснюється з ще залишившихся нерозподілених навантажень;
- відображення кількості залишившихся годин з дисциплін у навчальному плані з урахуванням вже розподіленого навантаження;
- швидкий пошук необхідної інформації по якомусь критерію;
- формування звіту на кожного викладача, загального звіту для всієї кафедри, а також звіту про нерозподілене навантаження. Звіти формуються у вигляді електронних таблиць MS Excel, які можна безпосередньо з програми переглянути і вивести на друк;
- резервне копіювання і відновлення даних;
- використання вбудованої довідкової системи.

Але ні по одній з даних систем немає повної інформації, яка дозволила б судити про її переваги і недоліки, які можуть також ховатися в НЕ ергономічності програми. Єдиний недолік, який видно з перерахованих функцій систем, це те, що перша система не підтримує виведення вихідних документів на друк, або імпорті цих документів у текстові або табличні редактори (наприклад, Microsoft Excel, Word).

2 ІНФОРМАЦІЙНО-ТЕХНОЛОГІЧНІ АСПЕКТИ

2.1 Розробка моделей автоматизованої системи

Оскільки кафедри, надають необхідну для формування навантаження інформацію, рознесені між інститутами університету, то для спрощення та прискорення процесу обміну даними між кафедрами зручно використовувати технології які вміють взаємодіяти с Microsoft Office. Узагальнена модель автоматизованої системи подана на рисунку 2.1.1.

Рис. 2.1.1. Узагальнена модель автоматизованої системи

Основними складовими системи є: ядро, база даних, інтерфейс користувача. Рис. 1. Узагальнена модель автоматизованої системи. База даних має нормалізовану структуру та зберігає усю необхідну інформацію про кафедри, викладачів, робочі плани, обрані студентами дисципліни. Ядро забезпечує обробку запитів користувача та здійснює взаємоузгоджене керування системними процесами. Інтерфейс користувача, створений за допомогою середовища Embercadero Delphi, має просту структуру, зручну та зрозумілу у використанні. Ядро системи забезпечує оперативну обробку вхідного інформаційного потоку, реалізує оновлення інформаційних ресурсів і формування вибірки даних із нормалізованої бази.

Методика автоматизованого розподілу навантаження дисциплін базується на реалізації розроблених моделей автоматизованої системи із забезпеченням вимог Болонського процесу щодо обов'язкового врахування у процесі формування навантаження блоків дисциплін вільного вибору студента.

2.2 Загальний опис моделювання

Одним з найбільш важливих видів системного аналізу є моделювання. Воно забезпечує опис ієрархічних відносин, механізмів і структур складної багатокомпонентної системи. Це дозволяє:

- візуалізувати систему в її нинішньому або очікуваному стані для проведення досліджень;
- визначити структуру системи та поведінку;
- отримати моделі, які можуть допомогти створити і моделювати комплекс системи.

Метод моделювання застосовується для формалізованого опису складної системи, що містить представлення елементарних об'єктів системи, їх особливості і відносини. На сьогоднішній день, разом з широко поширеною традицією використання методів аналітичного моделювання, велика увага приділяється методам моделювання системи, які дозволяють отримувати результати високої якості при аналізі складної динамічної системи. Вони дозволяють гнучко і очевидним чином відображати процеси, що відбуваються в складній системи.

Сьогодні, коли індустрія програмного забезпечення надає різні інструменти для моделювання, кожен висококваліфікований інженер і менеджер повинен вміти моделювати комплекс системи за допомогою сучасних технологій, реалізованих у вигляді графічно - інтегрованих середовищ і пакетів візуального моделювання.

2.3 Графічне моделювання

Моделювання бізнес-процесів, як правило, здійснюється та використовується бізнес-аналітиками і менеджерами, які прагнуть підвищити

ефективність процесу та їх якість. В крупних компаніях без формалізації і опису бізнес-процесів складно забезпечити належний рівень виконавської і технологічної дисципліни. Формалізація і опис бізнес-процесів є ключовою умовою для їх автоматизації. Взаємозв'язана система бізнеспроцесів зображає весь комплекс завдань і функцій структурних підрозділів, виконання яких необхідно забезпечити в процесі діяльності компанії. Моделювання бізнес-процесів дозволяє, незалежно від актуальної чисельності персоналу компанії, на будь-якому етапі її еволюційного розвитку, дозволяє закріпити ті або інші функції не тільки за конкретними структурними підрозділами, але і за конкретними фахівцями. В міру збільшення чисельності персоналу, створення нових структурних підрозділів можна гнучко перерозподіляти функції і завдання структурних підрозділів.

Графічний опис бізнес-процесів та їх імітація це методи аналізу бізнеспроцесів, ефективність яких доведена багаторічною практикою використання та численними дослідженнями.

2.4 Діаграми UML

Уніфікована мова моделювання (UML, Unified Modeling Language) є наступником методів об'єктно-орієнтованого аналізу і проектування (OOA & D), які з'явилися в кінці 80-х і початку 90-х років.

Перша згадка про уніфікованому методі (Unified Method) версії 0.8 з'явилося в 1995 році на конференції OOPSLA '95. Даний метод був запропонований Граді Бучом і Джимом Рамбо. Надалі до них приєднався Айвар Якобсон і в Протягом 1996 року М. Буч, Д. Рамбо, А. Якобсон, які отримали широку популярність як «троє друзів» (amigos) продовжували робота над своїм методом, який на той час отримав назву уніфікована мова моделювання (UML). Однак крім цього методу спільнотою розробником були запропоновані й інші методи. Для стандартизації цих методів в рамках OMG (Object Management Group) була сформована ініціативна група. В результаті роботи групи з'явилася версія мови UML.

UML - це назва мови моделювання, але не методу. Слід розрізняти ці поняття. Більшість методів включають в себе крім мови моделювання процес. Мова моделювання - це нотація (головним чином графічна), яка використовується розробниками для опису проекту. Процес - це рекомендація щодо етапів, яких необхідно дотримуватися при виконанні проекту.

UML - перш за все мова, і, як будь-який мовний засіб, він надає словник і правила комбінування слів в цьому словнику. В даному випадку словник і правила фокусуються на концептуальному і фізичному уявленнях системи. Мова диктує, як створити і прочитати модель, проте не містить ніяких рекомендацій про те, яку модель системи необхідно створити, - це виходить за рамки UML і є прерогативою процесу розробки програмного забезпечення.

В UML використовується два типи діаграм: структурні діаграми і поведінкові діаграми (рис. 2.4.1). Поведінкові діаграми представляють процеси, що відбуваються в середовищі, яке моделюється. Структурні діаграми являють собою елементи, які складають систему.

Є сім типів структурних діаграм, які повинні бути присутніми в будьякій системі, що моделюється:

- діаграма класів (Class Diagram);
- діаграма компонентів (Component Diagram);
- діаграми композитних структур (Composite Structure);
- діаграми розгортання (Deployment Diagram);
- діаграми об'єктів (Object diagram);
- діаграми пакетів (Package diagram).

Група поведінкових діаграм включає в себе:

- діаграми діяльності (Activity diagram);
- діаграми кінцевих автоматів (State machine diagram);
- діаграми варіантів використання (Use case diagram);
- діаграми взаємодії (Interaction overview diagram).

Підгрупа діаграм взаємодії контролює потік управління і потік даних, та включає в себе:

- діаграми послідовності (Sequence diagram);
- діаграми схем взаємодії (interaction overview diagrams);
- діаграми комунікацій (Comunication diagram);
- тимчасові діаграми (Timing diagram).

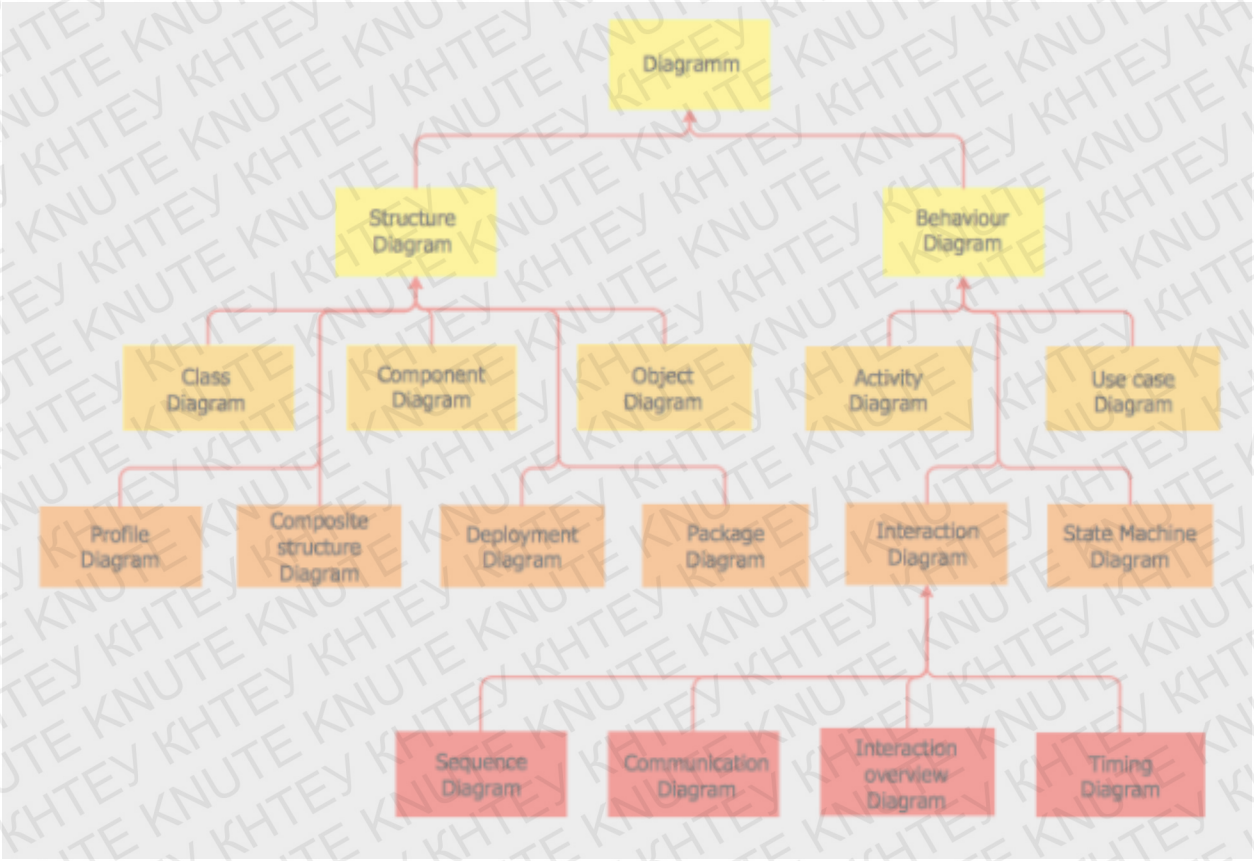


Рис. 2.4.1. Типи діаграм в мові UML

Діаграма прецедентів — діаграма, на якій зображено відношення між акторами та прецедентами в системі, також, перекладається як діаграма варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (англ. use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

На рисунку 2.4.2 зображено діаграму прецедентів для автоматизованої інформаційної системи розподілу педагогічного навантаження на кафедрі. На діаграмі зображено відношення між екторами та прецедентами.

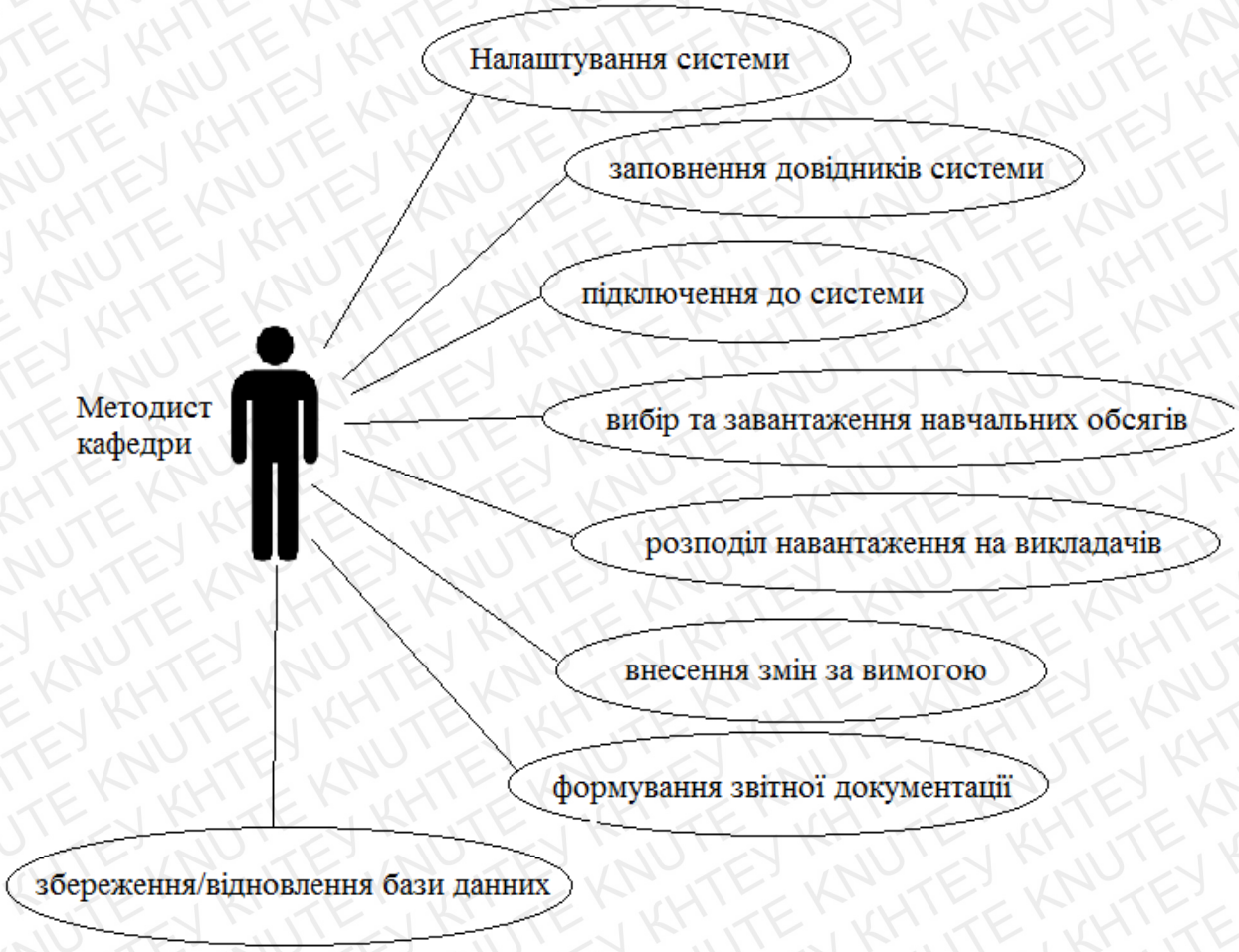


Рис. 2.4.2. Діаграма варіантів використання

Із системою взаємодії лише один актор – «Методист кафедри».

Прецеденти дають нам зрозуміти поведінку розроблюваної системи та отримати відповідь на запитання, що має робити система, але не визначають реалізацію цієї поведінки системи – не торкаються питань, яким чином відповідні функції реалізуються. Актор «Методист кафедри» виконує такі дії:

- налаштування системи;
- заповнення довідників системи (Дані про викладачів і т.д);
- підключення до системи;
- вибір та завантаження навчальних обсягів;
- розподіл навантаження на викладачів;
- внесення змін за вимогою;

- формування звітної документації;
- збереження/відновлення бази данних.

Діаграму можна деталізувати для кожного актора та детально розписати всі прецеденти які вони виконуватимуть.

2.5 Інструментальні засоби розробки програмного продукту

Embarcadero RAD Studio - середовище швидкої розробки додатків (RAD) фірми Embarcadero Technologies, яка працює під Windows.

Поточна версія Embarcadero RAD Studio 10.2 Tokyo об'єднує Delphi і C ++ Builder в єдину інтегровану середу розробки.

Інтегроване середовище розробки RAD Studio дозволяє розробникам писати код швидше і раціональніше за рахунок використання сучасних прийомів об'єктно-орієнтованого програмування в поєднанні з надійними програмними каркасами і багатством функцій середовища розробки. Підтримка Agile для груп розробників по всьому світу: раціональна розробка, реструктуризація вихідного коду і написання власного коду з використанням RAD Studio.

Використання оптимізованих компіляторів для всіх цільових платформ гарантує високу продуктивність і виключає час простою ваших пальців при кожній компіляції. Гібридне поєднання власних компіляторів Windows і глибокої інтеграції універсальної системи аналізу, трансформації та оптимізації програм LLVM для мобільних пристроїв забезпечує передові для галузі швидкості при компіляції для різних платформ на базі кращого компілятора для кожної з них.

Основні нові можливості RAD Studio 10.2 Tokyo:

- до складу Delphi включений компілятор додатків під Linux (Ubuntu Server (x64) (LTS 16.04) and RedHat Enterprise (V7));
- включена підтримка СУБД MariaDB.
- підтримка Android Nougat
- підтримка Windows 10 Anniversary Update
- підтримка MacOS Sierra

- підтримка IOS 10

В дипломному проекті використовулася мова Object Pascal.

Для розробки автоматизованної системи було обране середовище Embarcadero RAD Studio, так як воно дає змогу використовувати велику кількість існуючих бібліотек та компонентів для створення ядра та інтерфейсу користувача. Також Rad Sudio дає можливість використовувати технології, що надають можливість працювати з об'єктами Microsoft Office без зайвих зусиль.

На рисунку 2.5.1 показано середовище розробки.

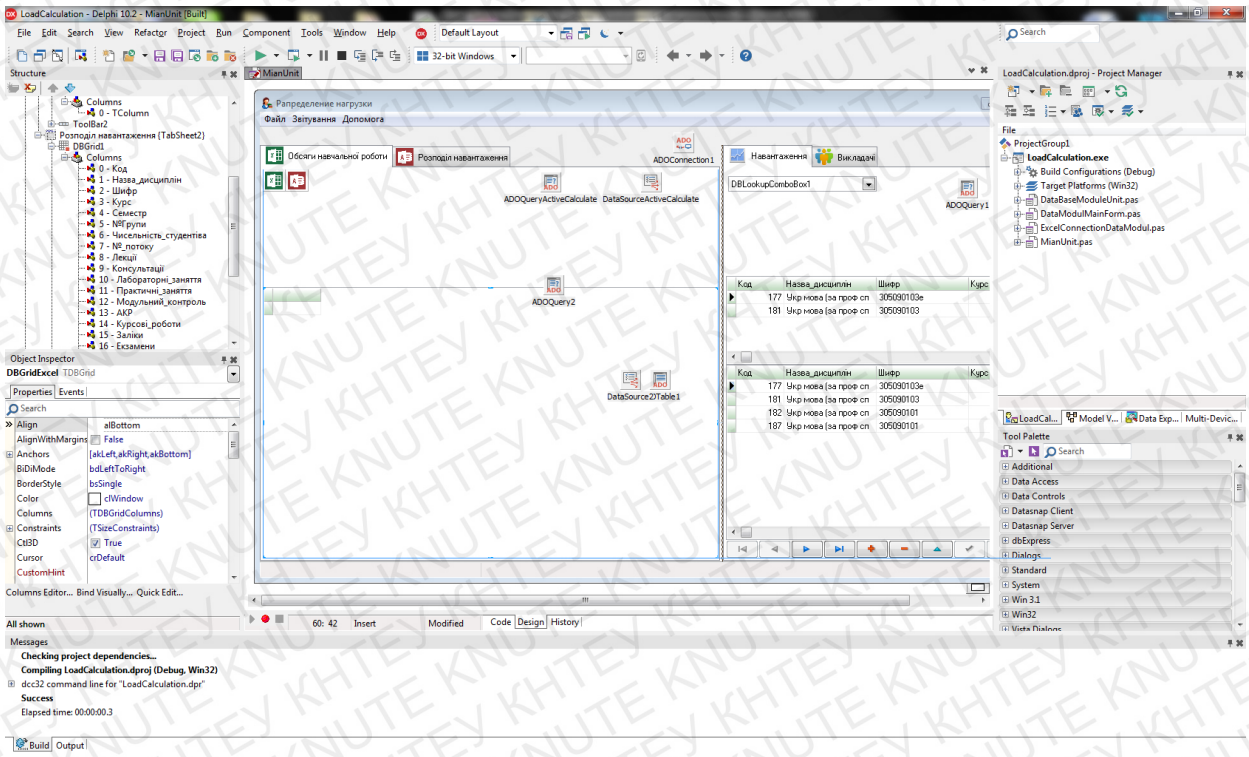


Рис. 2.5.1. Середовище розробки Embercadero Delphi 10.2

2.6 Характеристика мови програмування

Object Pascal — об'єктно-орієнтована мова програмування, нащадок Pascal, більш відома як основна мова програмування середовища Delphi. Мова програмування Object Pascal є останньою версією сімейства мов Паскаль, що реалізує принципи об'єктно-орієнтованого ного програмування. Ця мова є основою системи візуального програмування Delphi. Найбільш істотною відмінністю від традиційної мови Паскаль є наявність досить складних структур даних (класи) і можливість засобами Паскаль звертатися до функцій Windows API для створення повноцінних додатків Windows. Object Pascal

дозволяє використовувати безліч самих різноманітних типів і структур даних. Всі типи даних можна розбити на дві групи: прості (базові) та структуровані (користувальницькі) типи, які створюються на основі базових і об'єднують кілька змінних різних типів в одній структурі даних.

Класом у Object Pascal називається структура мови, що може мати у своєму складі перемінні, функції і процедури. Перемінні в залежності від призначення іменуються чи полями властивостями (див. нижче). Процедури і функції класу — методами. Відповідний класу тип будемо називати об'єктним типом:

```
type
TMyObject = class (Tobject)
MyField: Integer;
function MyMethod: Integer;
end;
```

У цьому прикладі описаний клас TMyObject, що містить поле MyField і метод MyMethod.

Поля об'єкта аналогічні полям запису (record). Це дані, унікальні для кожного створеного в програмі екземпляра класу. Описаний тут клас TMyObject має одне поле — MyField.

Методи — це процедури і функції, описані усередині класу і призначені для операцій над його полями. До складу класу входить покажчик на спеціальну таблицю, де міститься вся інформація, потрібна для виклику методів. Від звичайних процедур і функцій методи відрізняються тим, що їм при виклику передається покажчик на той об'єкт, що їх викликав. Тому оброблятися будуть поля саме того об'єкта, що викликав метод. Усередині методу покажчик на його об'єкт, що викликав, доступний під зарезервованим ім'ям Self.

Класи можуть бути описані або в секції інтерфейсу модуля, або на верхньому рівні вкладеності секції реалізації. Не допускається опис класів "де потрапило", тобто усередині процедур і інших блоків коду.

Дозволено випереджальне оголошення класів, як у наступному прикладі:

```
Type
```

```
TFirstObject = class;
TSecondObject = class(TObject)
Fist : TFirstObject;
…
end;
TFirstObject  = class(TObject)
...
end;
```

Щоб використовувати клас у програмі, потрібно, як мінімум, оголосити перемінну цього типу. Перемінна об'єктного типу називається екземпляром чи класу об'єктом.

```
var
AMyObject: TMyObject;
```

До введення терміна "клас" у мові Pascal існувала двозначність визначення "об'єкт", що міг позначати і тип, і перемінну цього типу. Тепер же існує чітка границя: клас — це опис, об'єкт — те, що створено відповідно до цього опису.

Як створюються і знищуються об'єкти?

Ті, хто раніш використовував ООП у роботі на C++ і особливо в Turbo Pascal, будьте уважні: у Object Pascal екземпляри об'єктів можуть бути тільки динамічними. Це означає, що в приведеному вище фрагменті перемінна AMyObject насправді є покажчиком, що містить адресу об'єкта.

Об'єкт "з'являється на світло" у результаті виклику спеціального методу, що инициализирует об'єкт — конструктора. Створений екземпляр знищується іншим методом — деструктором.

```
AMyObject := TMyObject.Create;
{ дії зі створеним об'єктом }
…
```

AMyObject...Destroy;

Зверніть увагу, що викликається метод TMyObject.create, а не AMyObject.Create. Є такі методи (у тому числі конструктор), що успішно працюють до (чи навіть без) створення об'єкта. Про подібні методи, називаних методами класу, піде мова трохи нижче.

У Object Pascal конструкторів у класу може бути кілька. Загально прийнято називати конструктор Create (на відміну від Turbo Pascal, де конструктор звичайно називався init, і від C++, де його ім'я збігається з ім'ям класу). Типова назва деструктора — Destroy.

type

```
TMyObject = class(TObject)
MyField: Integer;
Constructor Create;
Destructor Destroy;
Function MyMethod: Integer;
end;
```

Для знищення екземпляра об'єкта рекомендується використовувати метод Free, що спочатку перевіряє покажчик (чине дорівнює він їй) і тільки потім викликає Destroy:

```
AMyObject.Free;
```

До передачі керування тілу конструктора відбувається власне створення об'єкта — під нього приділяється пам'ять, значення всіх полів обнуляються. Далі виконується код конструктора, написаний програмістом для ініціалізації екземплярів даного класу. Таким чином, хоча на перший погляд синтаксис конструктора схожий з викликом процедури (не визначене значення, що повертається,), але насправді конструктор — це функція, що повертає створений і ініціалізуваний об'єкт.

Щоб правильно ініціалізувати у створюваному об'єкті поля, що відносяться до класу-предка, потрібно відразу ж при вході в конструктор викликати конструктор предка за допомогою зарезервованого слова inherited:

```
constructor TMyOb ject. Create;
begin
inherited Create;
…
end;
```

2.7 Компоненти доступу до даних в середовищі Delphi

Особливості компонентів доступу до даних.

Компоненти доступу до даних є невізуальними компонентами. Таблиці БД розташовуються на диску і є фізичними об'єктами. Для операцій з даними, що містяться в таблицях, використовуються набори даних. У термінах системи Delphi набір даних представляє собою сукупність записів, взятих з однієї або декількох таблиць БД. Записи, що входять в набір даних, відбираються за певними правилами, при цьому в окремих випадках набір даних може включати в себе всі записи з пов'язаної з ним таблиці або не містити жодного запису. Набір даних є логічною таблицею, з якою можна працювати при виконанні програми. Взаємодія таблиці і набору даних нагадує взаємодія фізичного файлу і файлової змінної.

У Delphi для роботи з наборами даних служать компоненти DBTable і ADOTable, DBQuery і ADOQuery, DataAccess, DataControl, DecisionQuery і StoredProc.

Компонент StoredProc використовується для виклику збережених процедур при організації взаємодії з віддаленими БД, а компонент UpDateSQL забезпечує роботу з кешуватися змінами в записах. Компонент DecisionQuery застосовується при побудові систем прийняття рішень. Найбільш універсальними і, відповідно, часто використовуваними є компоненти Table і Query, що задають набори даних.

Компонент доступу до даних DataSet.

Базові можливості доступу до БД забезпечує клас DataSet, що представляє набори даних у вигляді сукупності рядків і стовпців (записів і полів). Цей клас надає основні засоби навігації та редагування наборів даних.

Компонент DataSet призначений для подання набору даних зі сховища даних ADO. Він простий у використанні, маючи тільки кілька власних властивостей і методів.

Це єдиний компонент ADO, інкапсулює набір даних, для якого опубліковані властивості, що дозволяють управляти командою ADO. В результаті компонент являє собою гнучкий інструмент, який дозволяє (в

залежності від типу команди і її тексту) отримувати дані з таблиць, запитів SQL, збережених процедур, файлів і т. Д.

Компоненти доступу до даних Table і Query.

Компоненти Table і Query є похідними від класу ADODataSet нащадка класу DataSet. Вони демонструють схожі з базовими класами характеристики і поведінку, але кожен них має і свої особливості.

Компонент ADOTable забезпечує використання в додатках Delphi таблиць БД, підключених через провайдери OLE DB. За своїми функціональними можливостями і застосування він подібний до стандартного табличному компоненту. В основі компонента лежить використання команди ADO, але її властивості налаштовані заздалегідь і зміні не підлягають.

Інші властивості і методи компонента забезпечують застосування індексів. Так як не всі провайдери ADO забезпечують пряме використання таблиць БД, то для доступу до них може знадобитися запит SQL.

Компонент Table являє собою набір даних, який в деякий момент часу може бути пов'язаний тільки з однією таблицею БД. Цей набір даних формується на базі навігаційного способу доступу до даних, тому компонент Table рекомендується використовувати для локальних БД, таких як dBase і Paradox. При роботі з віддаленими БД слід використовувати компонент Query. Зв'язок між таблицею і компонентом Table встановлюється через його властивості TableName, яке задає ім'я таблиці.

При завданні значення властивості TableName вказується ім'я файлу і розширення ім'я файлу. На етапі розробки програми імена всіх таблиць доступні в списку Інспектора об'єктів. У цей список потрапляють таблиці, файли яких розташовані в каталозі, вказаному у властивості TableName.

Компонент ADOQuery забезпечує застосування запитів SQL при роботі з даними через ADO. За своєю функціональністю він подібний до стандартного компоненту запиту.

Компонент Query являє собою набір даних, записи якого формуються в результаті виконання SQL-запиту і засновані на реляционном способі доступу до даних. При роботі з віддаленими БД рекомендується використовувати набір

даних Query. При роботі з віддаленими базами даних слід звертатися до засобів мови SQL. Це відноситься і до таких операцій, як переміщення по набору даних або вставка в нього записів. Якщо ж для компонента Query використовуються методи типу Next і Insert, то замість реляційного способу доступу до даних буде застосований навігаційний. В результаті набору даних Query буде мало чим відрізнятися від набору даних Table. На відміну від компонента Table. Набір даних Query може включати в себе записи більш ніж однієї таблиці БД. Текст запиту, на підставі якого в набір даних відбираються записи, міститься у властивості SQL типу Strings. Запит включає в себе команди на мові SQL і виконується при відкритті набору даних. Запит SQL іноді називається SQL-програмою. SQL-запит можна формувати і змінювати динамічно, вносячи зміни в його текст безпосередньо при виконанні програми.

2.8 Компоненти для роботи з даними

Поняття і особливості візуальних компонентів в середовищі Delphi.

Візуальні компоненти для роботи з даними розташовані на сторінці DataControls - палітри компонентів і призначені для побудови інтерфейсної частини програми. Вони використовуються для навігації по набору даних, а також для відображення і редагування записів. Часто ці компоненти називаються елементами, чутливі до даних.

Одні візуальні компоненти для роботи з даними призначені для виконання операцій з полями окремого запису, вони відображають і дозволяють редагувати значення поля поточного запису. До таких компонентів належать, наприклад, однорядковий редактор Edit і графічний огляд Image.

Інші компоненти служать для відображення і редагування відразу декількох записів. Прикладами таких компонентів є сітки DBGrid і DBCtrlGrid, що виводять записи набору даних в табличному вигляді. Візуальні компоненти для роботи з даними схожі на відповідні компоненти сторінок Standard і Additional і відрізняються, в основному, тим, що орієнтовані на роботу з БД і мають додаткові властивості DataSource і Datafield. Перше з них вказує на

джерело даних, а друге - на поле набору даних, з яким пов'язаний візуальний компонент. Наприклад, Edit відображає строкове значення, дозволяючи користувачеві змінювати його.

Для всіх візуальних компонентів, призначених для відображення і редагування значень полів, при зміні користувачем їх вмісту набір даних автоматично переводиться в режим редагування. Зроблені за допомогою цих компонентів зміни автоматично зберігаються в пов'язаних з ними полях при настанні певних подій.

При програмному зміні вмісту ці візуальних компонентів набір даних автоматично в режим редагування не перекладається. Для цієї мети в коді повинен попередньо викликатися метод Edit набору. Щоб зберегти зміни в поле (полях) поточного запису, ми повинні також передбачити відповідні дії, наприклад, виклик методу Post або перехід до іншого запису набору даних.

Загальна характеристика візуальних компонентів.

У бібліотеці візуальних компонентів для всіх компонентів, в тому числі і призначених для роботи з даними, базовим є класу Control. Він забезпечує основні функціональні атрибути такі, як положення і розміри елемента, його заголовок, колір і інші параметри. Клас Control включає в себе загальні для візуальних компонентів властивості, події і методи. В цілому візуальні компоненти можна розділити на дві групи: віконні та не віконні.

Віконний елемент керування являє собою спеціалізоване вікно, призначене для вирішення конкретного завдання. До таких елементів відносяться, наприклад, поля редагування, командні кнопки, смуги прокрутки.

Такі компоненти, як Edit, DBEdit, Memo або DBMemo при отриманні фокусу вводу відображають в своїй області курсор редагування. Комплектуючих виробів, не пов'язані з редагуванням інформації, отримання фокусу введення зазвичай відображають за допомогою за допомогою пунктирного чорного прямокутника.

До неоконів елементам управління базовим є клас GraphiControl, вироблений безпосередньо від класу Control. Неоконів елементи управління на можуть отримувати фокус введення. Їх перевагою є менш витрачання ресурсів.

Властивості і події візуальних компонентів для роботи з даними.

Властивості дозволяють управляти зовнішнім виглядом і поведінкою компонентів при проектуванні і при виконанні програми. Зазвичай установка значень більшості властивостей компонентів виконується на етапі проектування за допомогою інспектора об'єктів.

Властивість Name вказує на ім'я компонента, яке використовується для управління компонентів під час виконання програми. Кожен новий компонент, що поміщається на форму, отримує ім'я за замовчуванням, автоматично утворене шляхом додавання до назви компонента його номера в порядку приміщення на форму. На етапі розробки програми ми можемо змінювати ім'я компонента на більш осмислене і відповідне призначенням компонента.

Властивість Aling визначає спосіб вирівнювання компонента на самій формі, на якій воно знаходиться. Вирівнювання використовується в разі, коли потрібно, щоб будь-якої інтерфейсний елемент займав певне положення.

Властивість Caption містить рядок для написи заголовка компонента. Окремі символи в заголовку можуть бути підкреслені, вони означають комбінації клавіш швидкого доступу.

Властивість Color визначає колір фону. Часто зручно задавати кольори за допомогою констант. Відображені колір залежить від параметрів відеокарти і монітора, в першу чергу від встановленого кольорового вирішення.

Візуальні компоненти здатні генерувати і обробляти досить велика кількість подій різних видів. До найбільш загальних груп подій можна віднести наступні дії:

- Вибір керуючого елемента;
- Переміщення покажчика миші;
- За допомогою кнопок клавіатури;
- Отримання і втрата керуючим елементом фокусу введення;
- Переміщення об'єктів методом drag

Існують і більш складні події, що вимагають передачі додаткових параметрів, наприклад подія, пов'язана з переміщенням покажчика миші, передає координати покажчика.

2.9 Мова запитів SQL

SQL символізує собою Структурований Мова Запитів. Це - мова, яка дає можливість створювати і працювати в реляційних базах даних, які є наборами зв'язаної інформації, що зберігається в таблицях.

Інформаційний простір стає більш уніфікованими. Це призвело до необхідності створення стандартної мови, яка могла би використовуватися у великій кількості різних видів комп'ютерних середовищ. Стандартна мова дозволить користувачам, які знають один набір команд, використовувати їх для створення, знаходження, зміни та передачі інформації - незалежно від того, чи працюють вони на персональному комп'ютері, мережевій робочій станції, або на універсальній ЕОМ. У нашому все більш і більш взаємопов'язаному комп'ютерному світі, користувач забезпечений такою мовою, має величезну перевагу у використанні та узагальненні інформації з ряду джерел за допомогою великої кількості способів. Елегантність і незалежність від специфіки комп'ютерних технологій, а також його підтримка лідерами промисловості в області технології реляційних баз даних, зробило SQL основним стандартною мовою.

Мова SQL призначена для маніпулювання даними в реляційних базах даних, визначення структури баз даних і для управління правами доступу до даних в багатокористувацької середовищі. Тому, в мову SQL в якості складових частин входять:

- мова маніпулювання даними (Data Manipulation Language, DML)
- мова визначення даних (Data Definition Language, DDL)
- мова керування даними (Data Control Language, DCL)

Це не окремі мови, а різні команди однієї мови. Такий поділ проведено тільки з точки зору різного функціонального призначення ЦИХ команд.

Мова маніпулювання даними використовується, як це випливає з його назви, для маніпулювання даними в таблицях баз даних. Він складається з 4 основних команд:

- SELECT (обрати)
- INSERT (вставити)
- UPDATE (оновити)
- DELETE (видалити)

Мова визначення даних використовується для створення та зміни структури бази даних і її складових частин - таблиць, індексів, представлений (віртуальних таблиць), а також тригерів і збережених процедур. Основними його командами є:

- CREATE DATABASE (створити базу даних)
- CREATE TABLE (створити таблицю)
- CREATE VIEW (створити віртуальну таблицю)
- CREATE INDEX (створити індекс)
- CREATE TRIGGER (створити тригер)
- CREATE PROCEDURE (створити збережену процедуру)
- ALTER DATABASE (модифікувати базу даних)
- ALTER TABLE (модифікувати таблицю)
- ALTER VIEW (модифікувати віртуальну таблицю)
- ALTER INDEX (модифікувати індекс)
- ALTER TRIGGER (модифікувати тригер)
- ALTER PROCEDURE (модифікувати збережену процедуру)
- DROP DATABASE (видалити базу даних)
- DROP TABLE (видалити таблицю)
- DROP VIEW (видалити віртуальну таблицю)
- DROP INDEX (видалити індекс
- DROP TRIGGER (видалити тригер)
- DROP PROCEDURE (видалити збережену процедуру)

Мова керування даними використовується для управління правами доступу до даних і виконанням процедур в багатокористувацької середовищі. Більш точно його можна назвати "мова управління доступом". Він складається з двох основних команд:

- GRANT (дати права)
- REVOKE (забрати права)

З точки зору прикладного інтерфейсу існують два різновиди команд SQL:

- інтерактивний SQL
- вбудований SQL

Інтерактивний SQL використовується в Спеціальних утиліт (типу WISQL або DBD), що дозволяють в інтерактивних режимі вводить запити з впровадженням команд SQL, посилати їх для виконання на сервер і отримувати результати в призначених для цього вікні.

Вбудований SQL використовується в прикладних програмах, дозволяючи їм посилати запити до сервера і Обробляти отримані результати, в тому числі комбінуючи set-орієнтований і record-орієнтований підходи.

Найбільш важливою командою мови маніпулювання даними є команда SELECT. Операція вибірки дозволяє Отримати всі терміни (записи) або частину термін однієї таблиці.

До логічним операторам відносяться відомі оператори AND, OR, NOT, що дозволяють виконувати різні логічні дії: логічне множення (AND, "перетин умов»), логічне додавання (OR, "об'єднання умов»), логічне заперечення (NOT, "заперечення умов") . У наших прикладах ми вже застосовували оператор AND. Використання ЦИХ операторів дозволяє гнучко "налаштувати" умови відбору записів. Оператор AND означає, що загальний предикат буде істинним тільки тоді, коли умови, пов'язані з "AND", будуть істинний.

Оператор OR означає, що загальний предикат буде істинним, коли хоча б одна з умов, пов'язаних з "OR", буде істинним. Оператор NOT означає, що загальний предикат буде істинним, коли умова, перед Яким варто цей оператор, буде помилковим. В одному предикате логічні оператори виконуються в наступному порядку: спочатку виконується оператор NOT, потім - AND і тільки після цього - оператор OR. Для зміни порядку виконання операторів дозволяється використовувати дужки.

Зміна порядку виводяться рядків (ORDER BY)

Порядок виведених термін може бути змінений за допомогою додаткової пропозиції ORDER BY в кінці SQL-запиту.

Ця пропозиція має вигляд:

ORDER BY <порядок рядків> [ASC | DESC]

Порядок термін може здаватися одним з двох способів:

- іменами стовпців
- номерами стовпців

Спосіб упорядкування визначається додатковими зарезервованим словами ASC і DESC. Способом за замовчуванням - якщо нічого не вказано - є упорядкування "по зростанню" (ASC). Якщо ж вказано слово "DESC", то упорядкування буде проводитися "за зменшенням".

2.10 Введення в OLE автоматизацію

Настільні додатки це текстові процесори, електронні таблиці і т.д.- призначені для підвищення продуктивності праці користувачів.

Програмованість додатків дозволяє використовувати їх послуги не тільки людям, а й іншим програмам. В результаті, замість того щоб залишатися інструментами лише для кінцевих користувачів, настільні додатки стають наборами інструментів для програмістів. Забезпечення програмованість вимагає стандартизації способу надання однією програмою своїх сервісів іншій програмі. Для реалізації такого типу взаємодії Microsoft використовує модель COM. Додатки забезпечують доступ до своїх сервісів через інтерфейси своїх COM-об'єктів, після чого цими сервісами може скористатися будь-який фрагмент коду, здатний викликати методи COM-об'єкта. Програмісти, таким чином, отримують можливість створювати додатки поверх функціональності, що надається наявним програмним забезпеченням. В COM такий стандартний спосіб забезпечення програмованість називається OLE-автоматизацією (OLE Automation).

Отже, розробник може зробити додаток програмованим, визначивши об'єкти (так звані, об'єкти автоматизації, Automation objects) і інтерфейси COM, методи яких будуть прямо відображатися на внутрішні функції програми.

Зазвичай, хоча і не обов'язково, для цих цілей використовується стандартний інтерфейс IDispatch, розроблений групою Microsoft Visual Basic. Необхідність в цьому інтерфейсі виникла на зорі OLE-автоматизації через те, що Visual Basic, будучи одним з найбільш поширених засобів створення сценаріїв для програмованих додатків, не підтримував можливості виклику методів звичайних COM-інтерфейсів з віртуальної таблицею. За традицією, з якою доводиться миритися програмістам на інших мовах, OLE-автоматизація більшості додатків і сьогодні реалізується за допомогою IDispatch. Цей інтерфейс в даний час підтримується Microsoft Word, Microsoft Excel і масою інших додатків.

Інтерфейс IDispatch спроектований таким чином, що клієнт з його допомогою може звертатися до довільної групі методів, передаючи будь-які необхідні параметри. Щоб це діяло, розробник об'єкта, що реалізує IDispatch, повинен визначити, які в точності методи будуть доступні. Це досягається визначенням диспетчерського інтерфейсу (dispatch interface), скорочено - діспінтерфейса (dispinterface). Кожен екземпляр стандартного інтерфейсу IDispatch (об'єкт може підтримувати кілька примірників одночасно) забезпечує доступ до певного діспінтерфейсу.

Інший, не менш важливим завданням IDispatch і діспінтерфейсов є забезпечення механізму пізнього зв'язування навіть при відсутності або недоступності бібліотеки типу. Клієнт в період виконання може вимагати від самого об'єкта інформацію про тип, при цьому в стандартизованном вигляді доступна вся необхідна для використання об'єкта інформація: імена і ідентифікатори властивостей і методів, типи параметрів і т.п. Після цього клієнт отримує можливість динамічно генерувати запити до об'єкта.

Найважливішим завданням клієнта, що використовує IDispatch, є маршалинга параметрів запитів. Нагадаємо, що маршалинга (упаковка параметрів для пересилання між процесами) звичайного COM-інтерфейсу з віртуальної таблицею виконується заступником і заглушкою (proxy, stub). В даному ж випадку, клієнт сам зобов'язаний виконати упаковку параметрів для методу діспінтерфейса в якусь стандартну форму, звану варіантом (variant), а

також розпакування з варіанта результатів виклику, повернутих методом. Варіант визначає стандартну форму подання кожного параметра і ідентифікатор типу параметра для всіх типів, які використовуються Visual Basic: коротке ціле, довге ціле, рядок символів і т.д. Програмісти на інших мовах тому повинні використовувати лише відомі Visual.

У повсякденному житті пакет Microsoft Office займає лідируючі позиції. Важко знайти такий комп'ютер, де він не встановлений. І погодьтеся, було б добре, щоб програма мала можливість виводити свій результат в табличний редактор Excel.

Дуже багато документів створюються і зберігаються в форматі електронних таблиць Microsoft Excel. Незважаючи на те, що ці таблиці мають можливості для автоматичної обробки документа, набагато приємніше працювати в звичному середовищі, яка володіє до того ж набагато більш розвиненими можливостями.

2.11 Технології ADO. Використання ADO в Delphi

Технологія Microsoft ActiveX Data Objects забезпечує універсальний доступ до джерел даних з додатків БД. Таку можливість надають функції набору інтерфейсів, створені на основі загальної моделі об'єктів COM і описані в специфікації OLE DB.

OLEDB являє собою набір спеціалізованих об'єктів COM, інкапсулюючих стандартні функції обробки даних, і спеціалізовані функції конкретних джерел даних і інтерфейсів, що забезпечують передачу даних між об'єктами.

Відповідно до термінології ADO, будь-яке джерело даних (база даних, електронна таблиця, файл) називається сховищем даних, з яким за допомогою провайдера даних взаємодіє додаток. Мінімальний набір компонентів програми може включати об'єкт з'єднання, об'єкт набору даних, об'єкт процесора запитів.

Технологія ADO і інтерфейси OLE DB забезпечують для додатків єдиний спосіб доступу до джерел даних різних типів. Наприклад, додаток, що використовує ADO, може застосовувати однаково складні операції і до даних,

що зберігаються на корпоративному сервері SQL, і до електронних таблиць, і локальним СУБД. Запит SQL, спрямований будь-якого джерела даних через ADO, буде виконаний (Рисунок 2.10.1).

Рис. 2.10.1. Схема доступа до данных через ADO

Об'єкти OLEDB створюються і функціонують так само, як і інші об'єкти COM. Кожному об'єкту відповідає ідентифікатор класу CLSID, що зберігається в системному реєстрі. Для створення об'єкта використовується метод CoCreateinstance і відповідна фабрика класу. Об'єкту відповідає набір інтерфейсів, до методів яких можна звертатися після створення об'єкта.

В результаті додаток звертається не прямо до джерела даних, а до об'єкта OLEDB, який "вміє" подати дані (наприклад, з файлу електронної пошти) у вигляді таблиці БД або результату виконання запиту SQL.

Технологія ADO в цілому включає в себе не тільки самі об'єкти OLEDB, але і механізми, що забезпечують взаємодію об'єктів з даними і додатками. На цьому рівні найважливішу роль відіграють провайдери ADO, які координують роботу додатків з сховищами даних різних типів.

Така архітектура дозволяє зробити набір об'єктів і інтерфейсів відкритим і розширюваним. Набір об'єктів і відповідний провайдер може бути створений для будь-якого сховища даних без внесення змін у вихідну структуру ADO. При цьому істотно розширюється саме поняття даних - адже можна розробити набір об'єктів і інтерфейсів і для нетрадиційних табличних даних. Наприклад, це можуть бути графічні дані геоінформаційних систем, деревовидні структури з системних реєстрів, дані CASE-інструментів і т. д.

Так як технологія ADO заснована на стандартних інтерфейсах COM, які є системним механізмом Windows, це скорочує загальний обсяг працюючого програмного коду і дозволяє передавати програми БД без допоміжних програм і бібліотек.

У програмному забезпечені створена форма в якій знаходяться майже всі ADO компоненти (рисунок 2.10.2).

Рис. 2.10.2. DataModule форма з компонентами

2.12 Підсистема бази даних

База даних – це інструмент, використовуючи який, можна збирати й упорядковувати інформацію. У базах даних можна зберігати відомості про людей, продукти, замовлення тощо. Чимало баз даних створюються як список у текстовому редакторі або електронній таблиці. Цей список постійно збільшується, тому в даних з’являється дедалі більше повторів і

невідповідностей. Дані стає важко розуміти у вигляді списку. Крім того, так не дуже зручно шукати й видобувати підмножини даних, які потрібно переглянути. Коли у вас почнуть виникати ці проблеми, радимо перенести дані в базу даних, створену в системі керування базою даних (СКБД), наприклад Access.

Комп'ютеризована база даних – це контейнер об'єктів. Одна база даних може містити кілька таблиць. Наприклад, система відстеження запасів, у якій використовуються три таблиці, – це не три, а одна база даних, що містить три таблиці. Таблиці бази даних Access (крім спеціальних баз даних, у яких використовуються дані або код з іншого джерела) зберігаються в одному файлі з іншими об'єктами, такими як форми, звіти, макроси й модулі. Бази даних, створені у форматі Access 2007 (який також використовується в Access 2016, Access 2013 і Access 2010), мають розширення файлу ACCDB, а бази даних, створені в попередніх форматах Access, – MDB. Створювати файли в попередніх форматах файлів (наприклад, Access 2000 і Access 2002–2003) можна в Access 2016, Access 2013, Access 2010 або Access 2007.

Використовуючи Access, можна:

- додавати нові дані до бази даних, наприклад новий елемент до запасів;
- редагувати наявні дані в базі даних, наприклад змінювати поточне розташування елемента;
- видаляти відомості, якщо, наприклад, елемент продано або вилучено;
- упорядковувати й переглядати дані різними способами;
- надавати спільний доступ до даних іншим, використовуючи звіти, повідомлення електронної пошти, інтромережу чи Інтернет.

Таблиця бази даних схожа на електронну таблицю – в обох дані зберігаються в рядках і стовпцях. Тому зазвичай досить легко імпортувати електронну таблицю в таблицю бази даних. Головна відмінність між тим, як дані зберігаються в електронній таблиці та базі даних, – це спосіб, яким упорядковуються дані.

Щоб база даних була максимально гнучка, дані має бути впорядковано в таблиці, щоб позбутися зайвих елементів. Наприклад, зберігаючи відомості про працівників, слід настроїти відповідну таблицю, у яку дані кожного працівника потрібно ввести лише раз. Дані про продукти зберігатимуться у власній таблиці, а дані про філіали – в іншій. Ця процедура називається нормалізацією.

Кожен рядок у таблиці називається записом. У записах зберігаються окремі елементи даних. Кожен запис складається з одного або кількох полів. Поля відповідають стовпцям у таблиці. Наприклад, можна створити таблицю "Працівники", де кожен запис (рядок) містить відомості про окремого працівника, а поля (стовпці) містять дані різного типу, наприклад ім’я, прізвище, адресу тощо. Полям має бути призначено певний тип даних (текст, дата або час, число чи інший тип).

Основною передумовою розробки систем, що використовують бази даних, є прагнення об’єднати всі дані, які обробляються на підприємстві, в єдине ціле та забезпечити контрольований доступ до них. Сучасні потужні підприємства часто мають велику кількість підрозділів, які можуть фізично розташовуватись за сотні та навіть тисячі кілометрів один від одного.

Кожний з цих підрозділів має свою локальну базу даних. Якщо ці бази мають різні архітектури та використовують різні протоколи зв’язку, то їх розглядають як інформаційні острови, важкодоступні для інших. В такому випадку виникає необхідність об’єднати розрізнені бази даних в одне логічне ціле, тобто створити розподілену базу даних.

Розподілена база даних – це сукупність логічно зв’язаних баз даних або частин однієї бази, які розпаралелені між декількома територіально-розподіленими ПЕОМ і забезпечені відповідними можливостями для управління цими базами або їх частинами. Тобто, розподілена база даних реалізується на різних просторово розосереджених обчислювальних засобах, разом з організаційними, технічними і програмними засобами її створення і ведення.

В дійсності розподілена база даних є віртуальною базою даних, компоненти якої фізично зберігаються на декількох різних реальних базах даних на декількох різних вузлах.

На рис. 2.11.1 наведено приклад типової топології розподіленої бази даних.

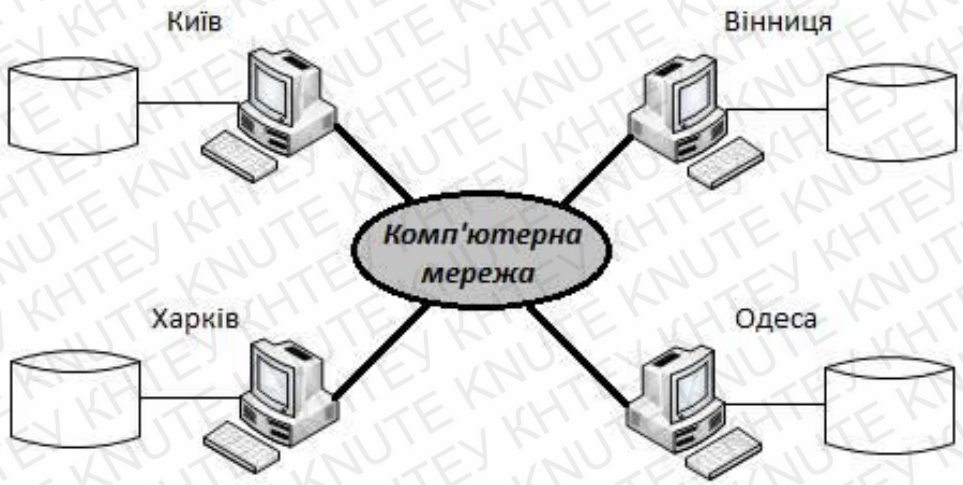


Рис. 2.11.1 Приклад типової топології розподіленої бази даних

У автоматизованій системі розподілу педагогічного навантаження була створена своя база даних, яка має наступну схему даних:

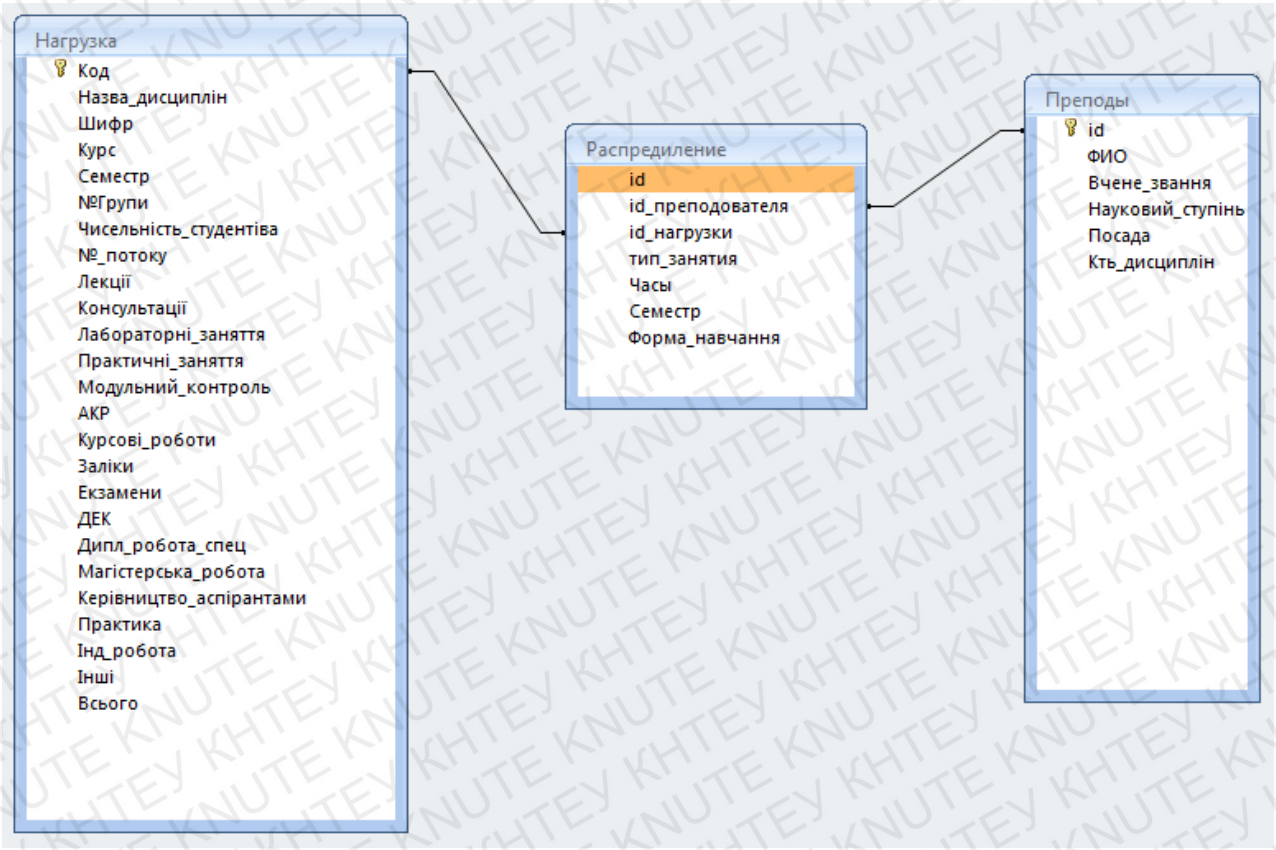


Рис. 2.11.2. Схема даних дипломного проекту

База даних складається з трьох таблиць:

- Навантаження
- Розподіл
- Викладачі

В таблиці "навантаження" зберігається вся інформація що надходить з обсягів на кафедру (рисунок 2.11.3).

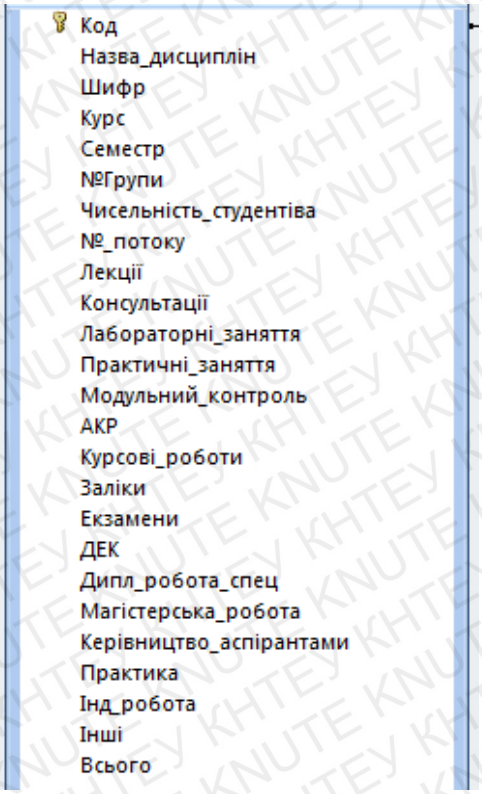


Рис. 2.11.3. Таблиця навантаження

Таблиця "розподіл" обєднує інші дві таблиці, по номеру навантаження та номеру викладача (рисунок 2.11.4).

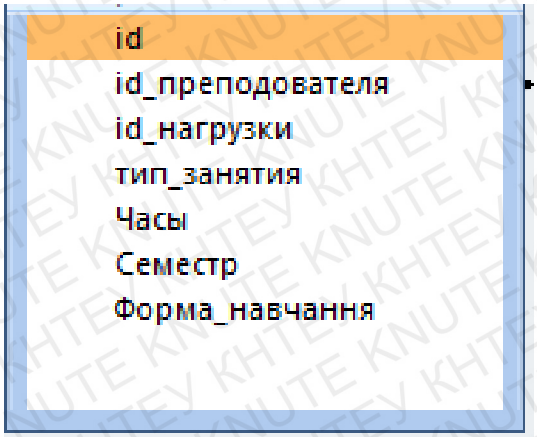


Рис. 2.11.4. Таблиця розподіл

3 ОПИС РОЗРОБЛЕНОЇ СИСТЕМИ РОЗПОДІЛУ ПЕДАГОГІЧНОГО НАВАНТАЖЕННЯ

3.1 Функціональне призначення

Однією з актуальних проблем роботи кафедр навчального закладу є розподіл і облік виконання навчального навантаження викладацького складу. Виконання цієї роботи займає багато часу, неминучі помилки і численні коригування.

Автоматизація данного процесу позбавляє співробітників кафедри від шаблонної роботи та дозволить:

- виключити багаторазове переписування одних і тих же д- не виробляти розрахунки, які не потребують уваги завідувача кафедри;
- скоротити час, займаний розрахунком;
- виключити ймовірність технічних помилок;
- оперативно реагувати на будь-які зміни, що стосуються навантаження.

Це допоможе підвищити ефективність дій у навчальній та науковій сферах. Внаслідок цього, розробка автоматизованої системи розрахунку педагогічного навантаження є актуальною.

Перед початком кожного навчального року на кафедру надходить план обсягів навчальної роботи кафедри (Рис. 3.1.1) , який має досить складну структуру.

Обсяги навчальної роботи кафедри на 2015/2016 н.р

Кафедра -35- Сучаснихних європейських мов (укр мова)

Назва навчальних дисциплін та робіт	Шифр навчального плану	Курс	Семестр	№ групи	Чисельність студентів	№ потоку (для лекцій)	№ потоку (для практичних)	Розрахунок годин																										
								Лекції	Практичні заняття	Лабораторні заняття	Модульний котроль	Семестровий контроль	АКР	Курсові роботи (проекти)	Лекції	Консультації	Лабораторні заняття	Практичні заняття	Модульний контроль	АКР	Курсові роботи (проекти)	Заліки	Екзамени (разом з консультацією перед ДЕК (бак., спец., маг.)	Дипл робота спец.	Магістерська робота	Керівництво аспірантами, наукове консультування	Практика	Інд робота (магістр)	Інші	Всього					
назва	план	кур	сем	гр	ст	пот.	пв	лек	прк	лаб	пмк	ісп	кон	кр	лек	конс	лаб	прк	мк	АКР	КР	зал	екз	ДЕК	дрс	мр	асп	пкт	інд	інш	всього				
Перше півріччя																																		
Денна форма																																		
обліку, аудиту та економічної кібернетики																																		
Укр мова (за проф спрямуванням)	30502	2	1	1	25	3145		16	14			3						14				2								16				
Укр мова (за проф спрямуванням)	30502	2	1	2	18	3145		16	14			3						14				2								16				
Укр мова (за проф спрямуванням)	305090101	2	1	3	19	3145		16	14			3			16			14				2								32				
Укр мова (за проф спрямуванням)	305090101	2	1	4	19	3145		16	14			3						14				2								16				
Укр мова (за проф спрямуванням)	305090103	2	1	5	22	3145		16	14			3						14				2								16				
Укр мова (за проф спрямуванням)	305090103	2	1	6	19	3145		16	14			3						14				2								16				
Укр мова (за проф спрямуванням)	50103	2	1	7	17	3145		16	14			3						14				2								16				
Всього по факультету															16			98				14								128				

Рис. 3.1.1. План обсягів навчальної роботи кафедри

У плані вказується:

- На якому курсі читається дисципліна
- З якої спеціальності
- Кількість груп на курсі
- Кількість тижнів
- Загальна кількість годин на всі наявні види занять (лекції, лабораторні та практичні роботи, заліки, іспити, контрольні і т.д.).

Завідувач кафедри на підставі цього документа розподіляє навантаження між співробітниками кафедри. Це одна з відповідальних і досить трудомістких обов'язків завідувача кафедри ВНЗ, особливо якщо велике число дисциплін, що читаються на кафедрі, а кадровий склад на кафедрі досить численний і динамічний.

У системі буде здійснюватись розрахунок, як загального навантаження кожного викладача, так і їх лекційної навантаження.

До складу вихідних документів включені наступні:

- деталізовані і зведені дані щодо розподілу навчального навантаження;
- відомості для індивідуальних планів викладачів;

Розроблена програма дозволяє формувати вихідну документацію - звіт про навантаженні окремо взятого викладача. Ним є документ Microsoft Excel, що містить таблицю навантаження.

Для автоматизації розрахунку необхідно в якості вхідної інформації зберігати дані з усіх навчальних планів, які реалізуються в поточному навчальному році, навчальним групам, "прикріпленим" до цих планів, нормам часу на виконання окремих видів робіт та відомості про викладачів.

Вхідними даними при розподілі навчального навантаження викладачів є:

- дані про закріплені за кафедрою дисципліни, а також групи, в яких ці дисципліни повинні читатися.
- дані про позааудиторні навантаження, не пов'язані з навчальними дисциплінами (практики, ДАК, завідування кафедрою);
- межі допустимого інтервалу зміни річного навчального навантаження (в годинах) за посадами;
- список викладачів кафедри із зазначенням посад.

Схематичне представлення роботи системи представлено на Рис.3.1.2.

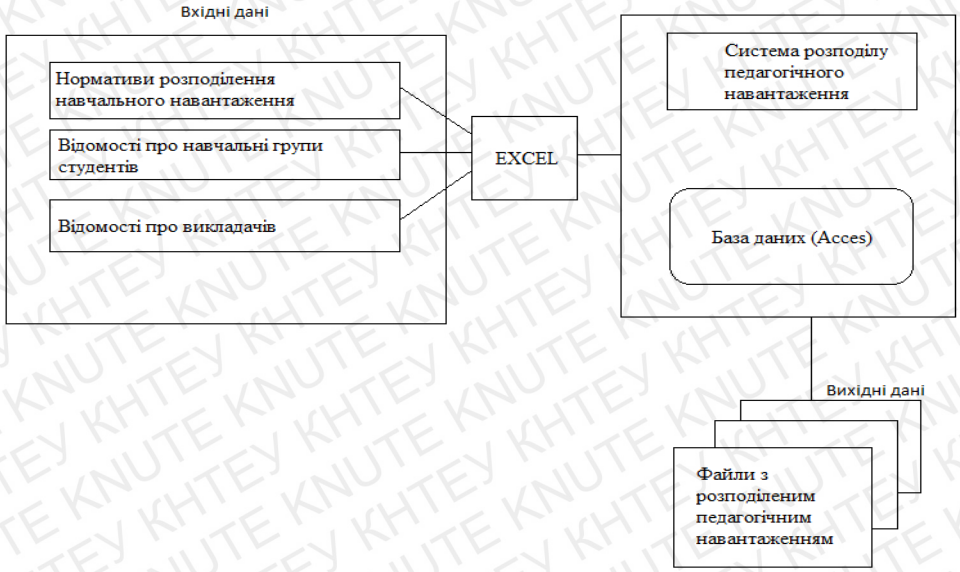


Рис. 3.1.2. Схема роботи автоматизованої системи розподілу педагогічного навантаження

3.2 Вимоги до програмного продукту

Для того щоб програма могла виповняти свої функції, Пк користувача повинен мати наступні програми:

- Microsoft Excel
- Microsoft Access

Програма була написана та перевірена на комп'ютері з наступними характеристиками (Рисунок 3.2.1):

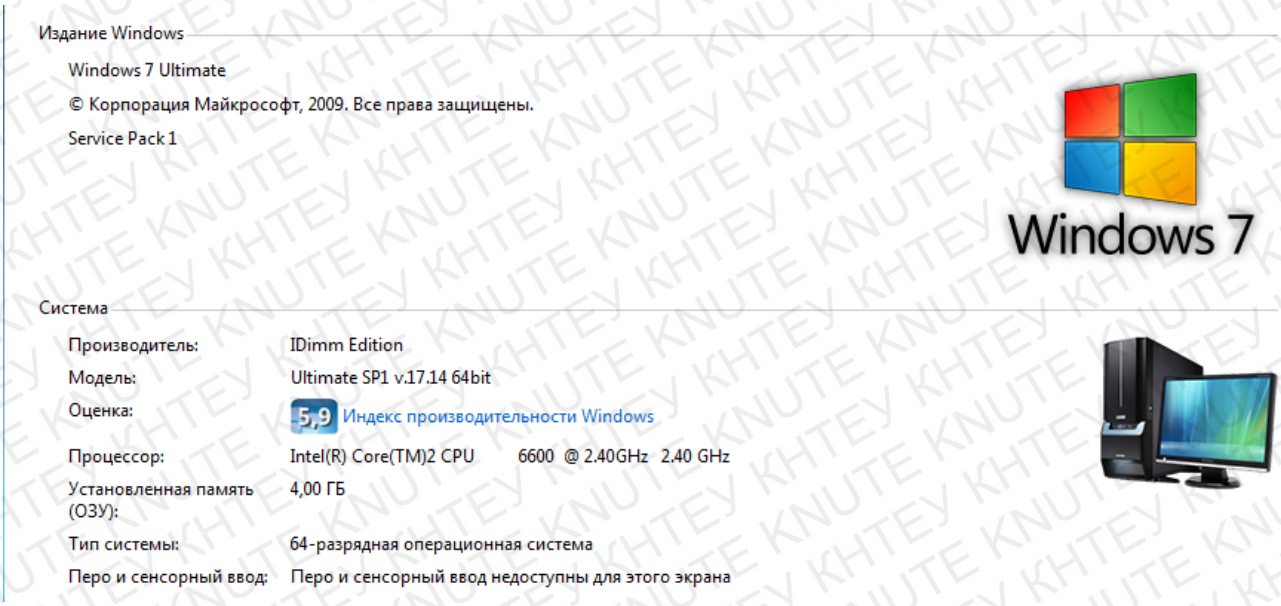


Рис. 3.2.1. Характеристики

Комп'ютер повинен бути оснащений наступними пристроями введення/виведення:

- Комп'ютерна миш
- Клавіатура
- Принтер

Наступне програмне забезпечення також повинно бути встановлено на ПК:

- Microsoft Exec
- Microsoft Access

3.3 Тестовий запуск програми

Відкривається головна форма програми (рисунок 3.3.1). На ній ми можемо побачити наступні елементи:

- Головне меню
- Вкладка "обсяги навчальної роботи"
- Вкладка "розподіл навантаження"
- Вкладка "Навантаження"
- Вкладка "Викладачі"

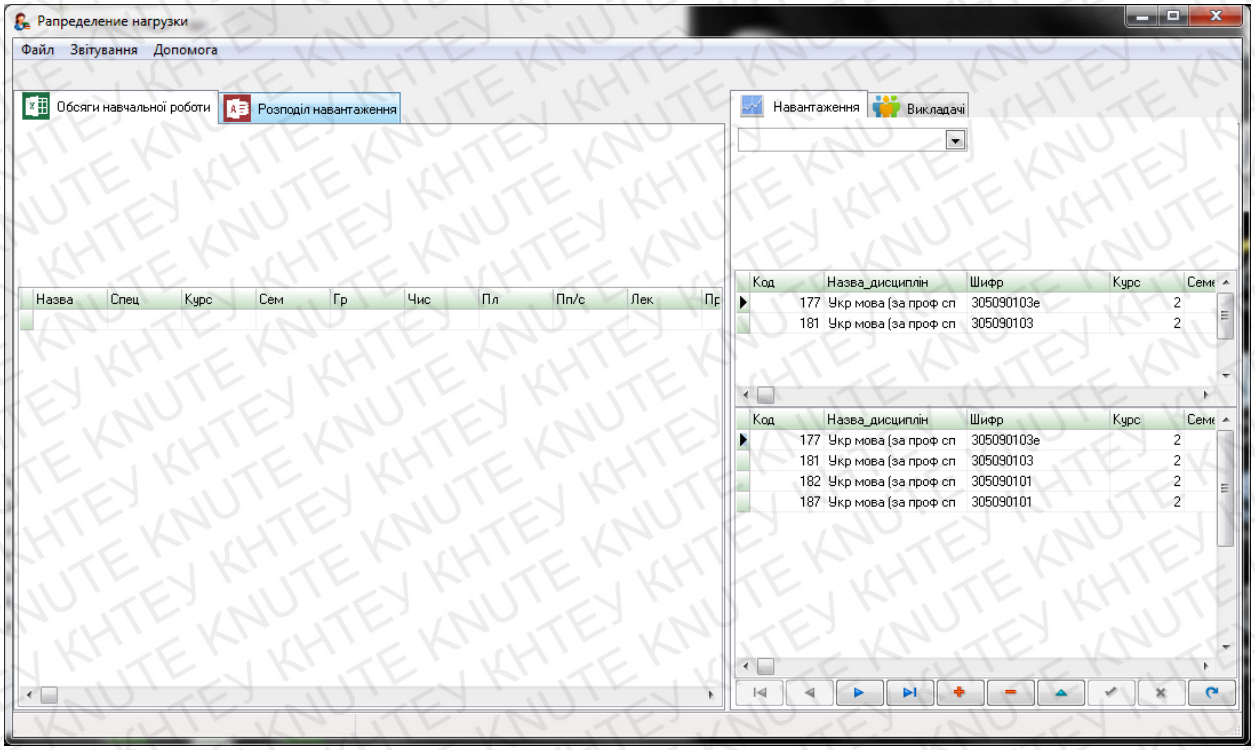


Рис. 3.3.1. Головна форма програми

Візуально програма складається з двох частин. У лівій половоні відображається обраний файл з обсягами навчальних робіт, та можливі маніпуляції з ними:

- Зберегти обсяг
- Видалити обсяг
- Змінити обсяг

Для маніпуляцій з данними котрі вже записані в базу даних, обравши вкладку "розподіл навантаження", зявляється можливість продивлятися навантаження, розподіляти їх на викладачів.

В правій частині знаходиться панель та таблиці з інформацією по навантаженню на викладачів. Їх группи, предмети, сумма годин і т.д.

Та вкладка з викладачами, де виконується редагування, додавання, видалення данних по них (рисунок 3.3.2).

Рис. 3.3.2. Вкладка викладачі

У нижній частині розташована панель навігації по данним:

- Перейти до першого запису
- Попередній запис

- Наступний запис
- Остній запис
- Додати викладача
- Видалити викладача
- Редагувати дані
- Зберегти дані
- Відмінити зміни
- Оновити таблицю

Вибравши в головному меню "Файл - Відкрити обсяги" (рисунок 3.3.3), з'явиться діалогове вікно де користувачу потрібно вибрати файл (рисунок 3.3.4). Існує декілька умов для правильної роботи з файлом:

- Файл повинен бути формату Excel документа
- Файл повинен мати один лист (листи крім першого будуть ігноруватися)
- Файл повинен мати правильну структура даних

Детальніше про умови роботи з вхідними файлами буду описано далі.

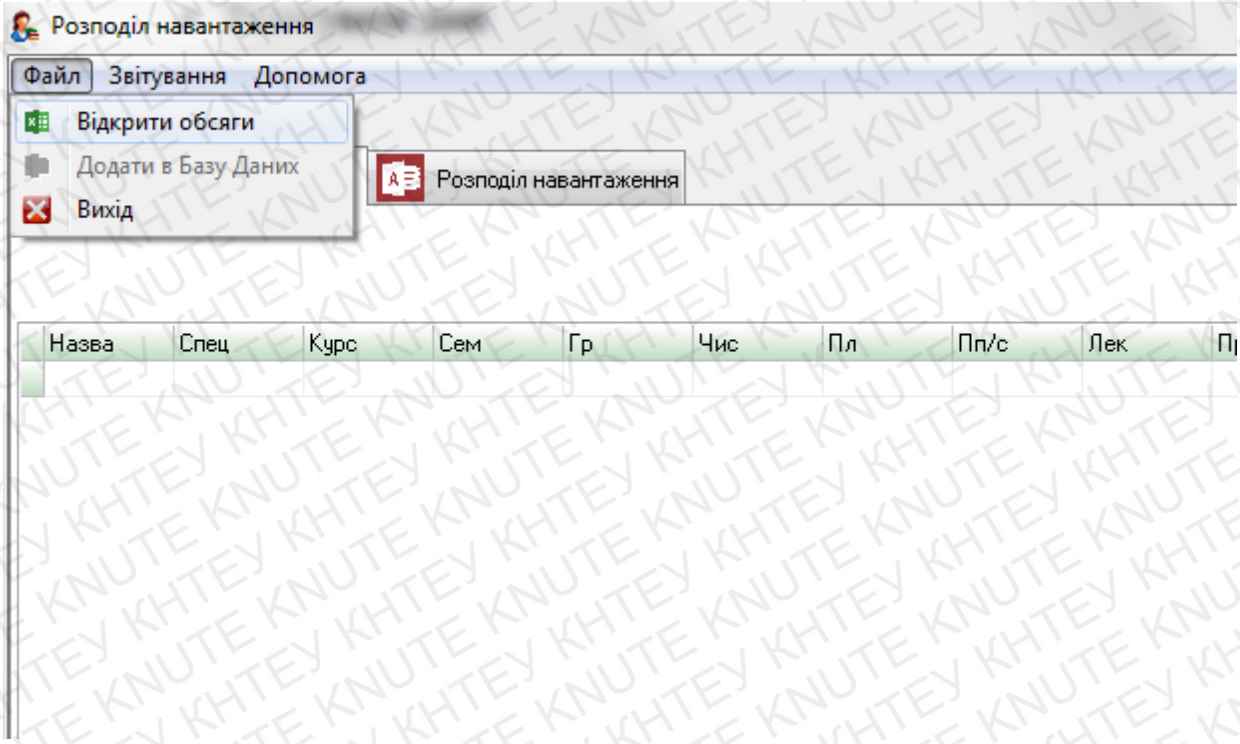


Рис. 3.3.3. Головна форма програми

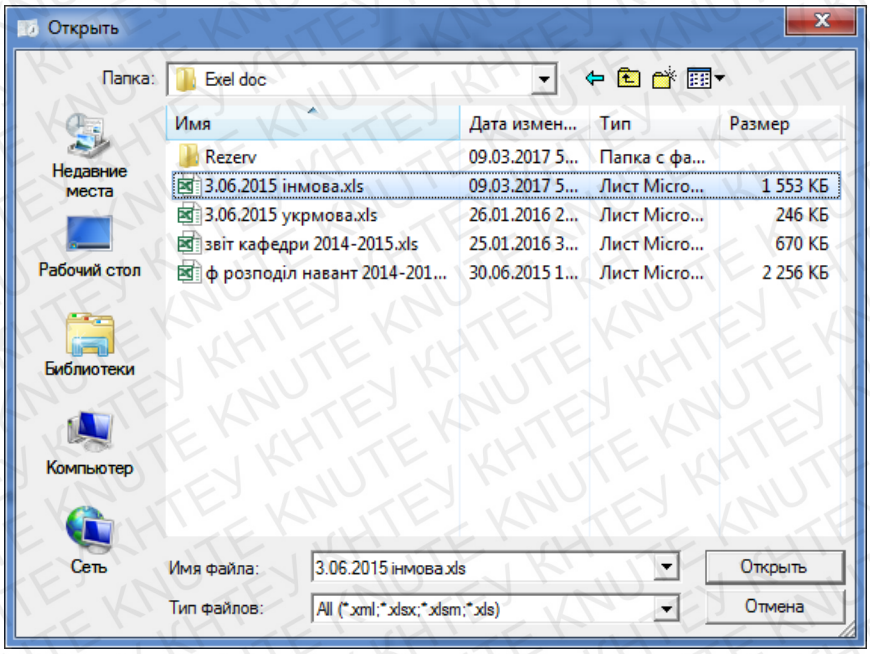


Рис. 3.3.4. Вибір файлу

Якщо були виконані все умови вхідних даних, то дані з файлу зчитаються і з’являться в першій таблиці (рисунок 3.3.5).

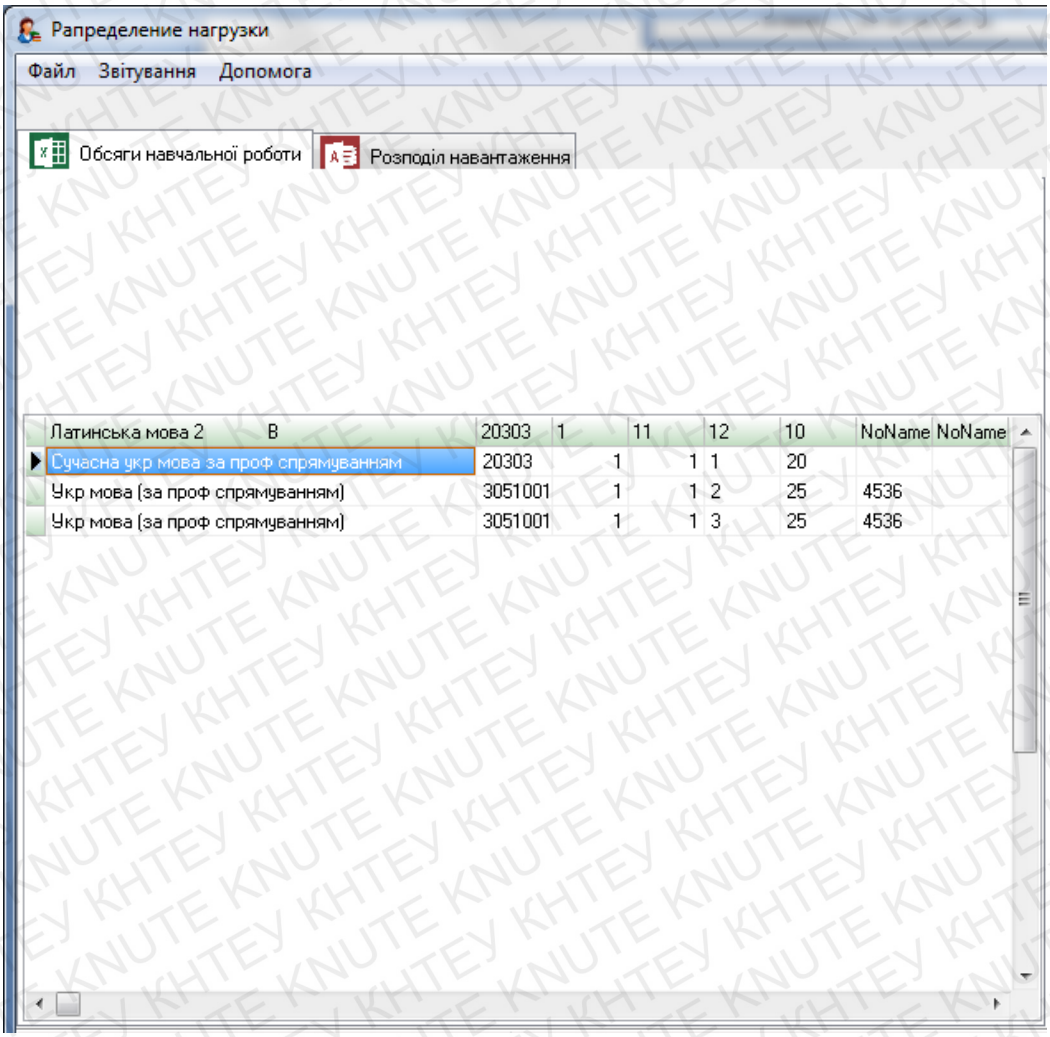


Рис. 3.3.5. Вхідні дані

Після прегляду даних та переконання в їх коректності, користувач має можливість зберегти інформацію в базу даних для подальшого її використання. Для цього в розділі "Файл" обираємо "Додати в Базу Даних" (рисунок 3.3.6).

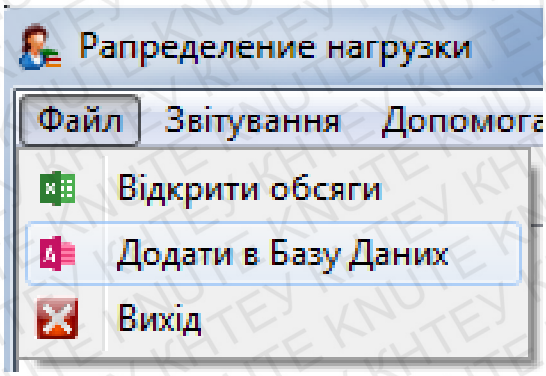


Рис. 3.3.6. Додавання обсягів в базу даних

Розподіл визначається користувачем обравши (рисунок 3.3.7).:

- Викладача (котрому обирається навантаження)
- Вид заняття (Лекції, Практичні, Лабораторні)
- Форма навчання(Денна, Вечірня, Заочна)

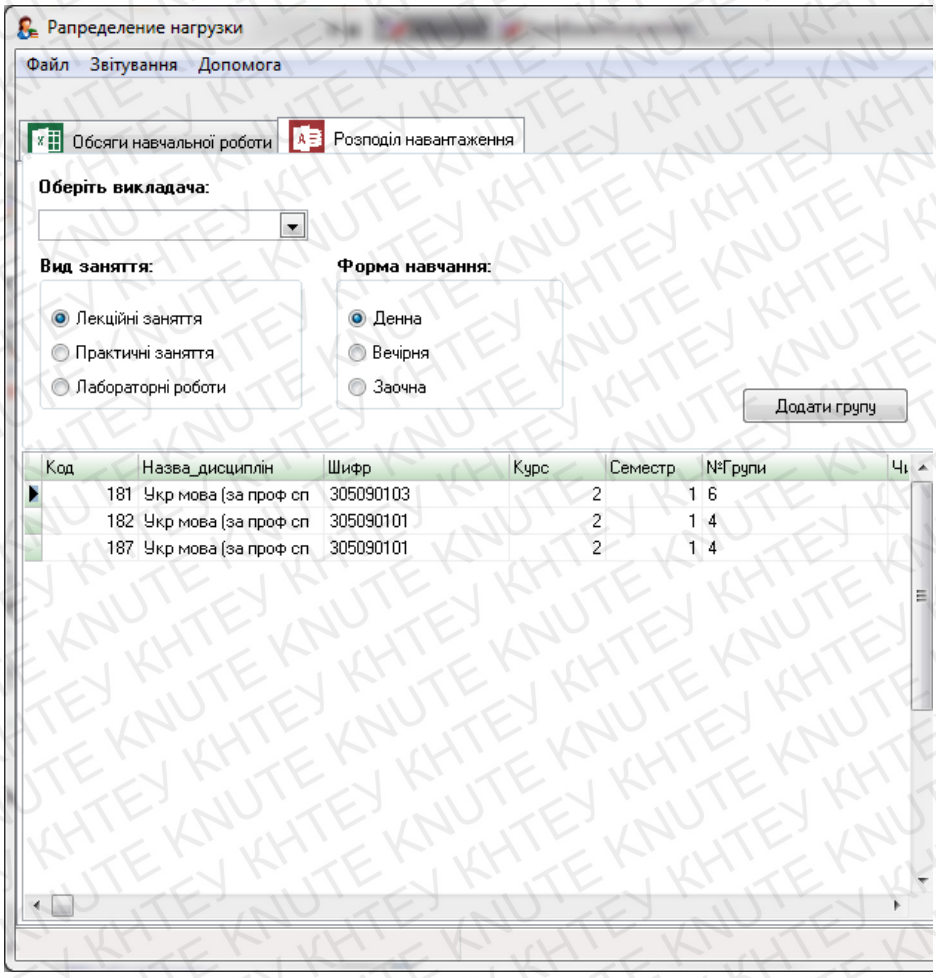


Рис. 3.3.7 Розподіл груп на викладачів

Користувач має можливість маніпулювати даними що вже розподілив, в вкладці "Навантаження", за допомогою панелі навігації по записам(Рисунок 3.3.8).

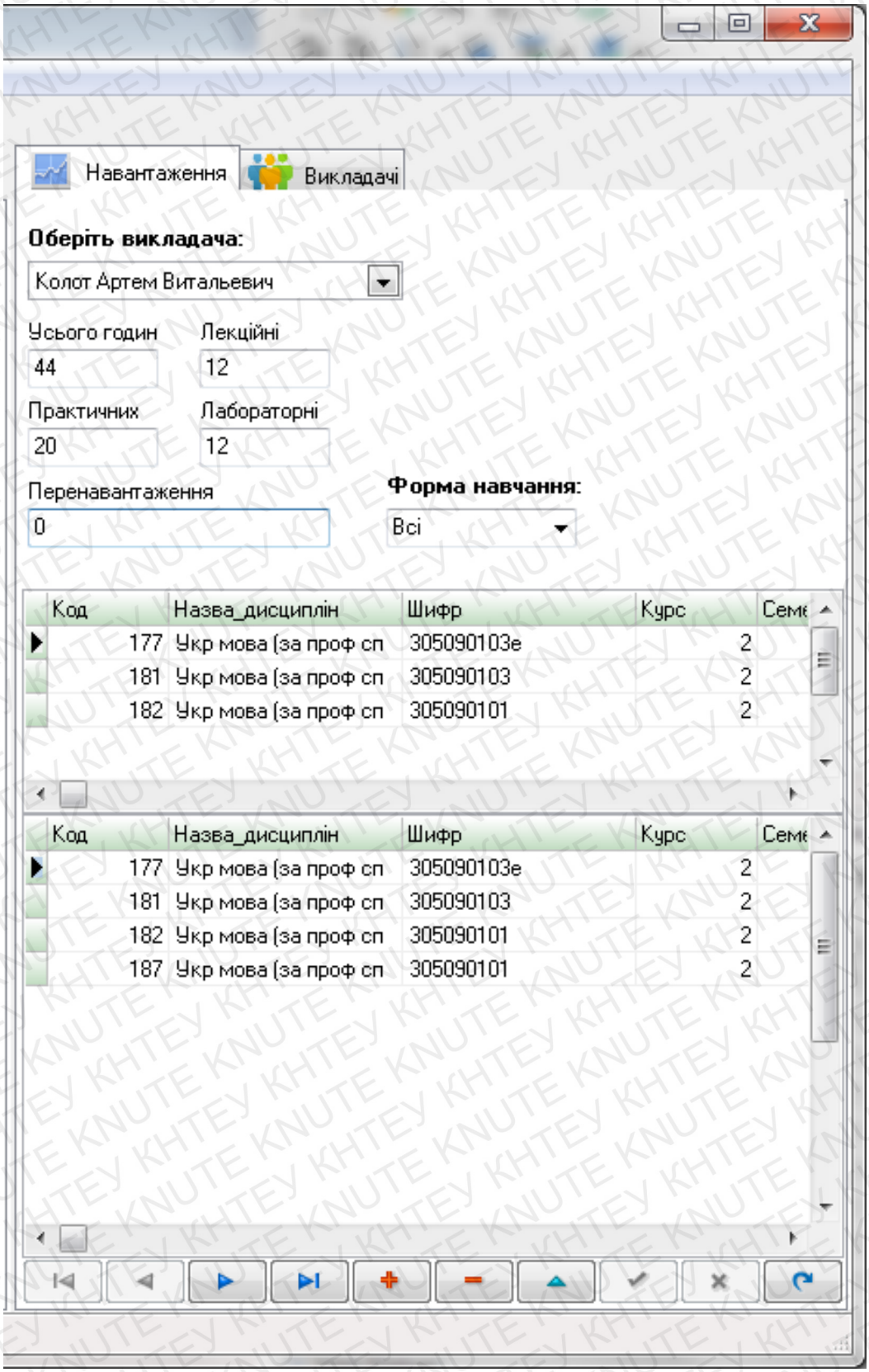


Рис. 3.3.8. Маніпулювання даними котрі вже розподілені

Можливі наступні дії:

- Перейти до першого запису
- Попередній запис
- Наступний запис
- Остній запис
- Видалити запис

Також ведеться статистика викладача, у полях записується сумма годин, сумма лекційних годин, сумма практичних годин, сумма лабораторних годин, та повідомляється про перенавантаження на викладача (рисунок 3.3.9).

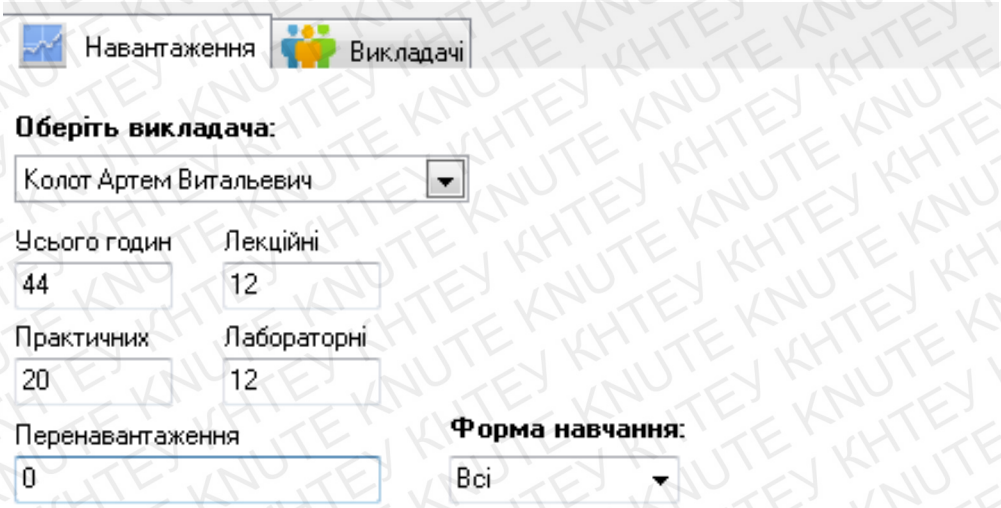


Рис. 3.3.9. Статистика

За допомогою розділу "Звітування" уголовному меню, існує можливість створювати табелі з розроділенним навантаженням (рисунок 3.3.10).

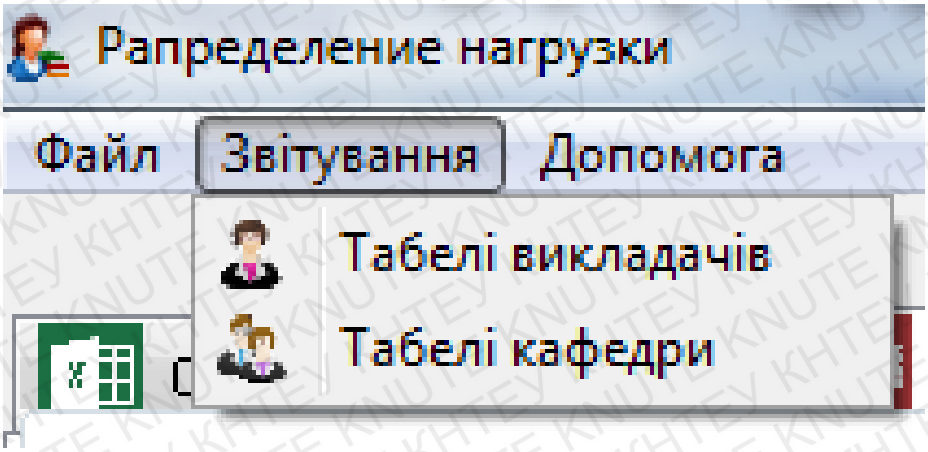
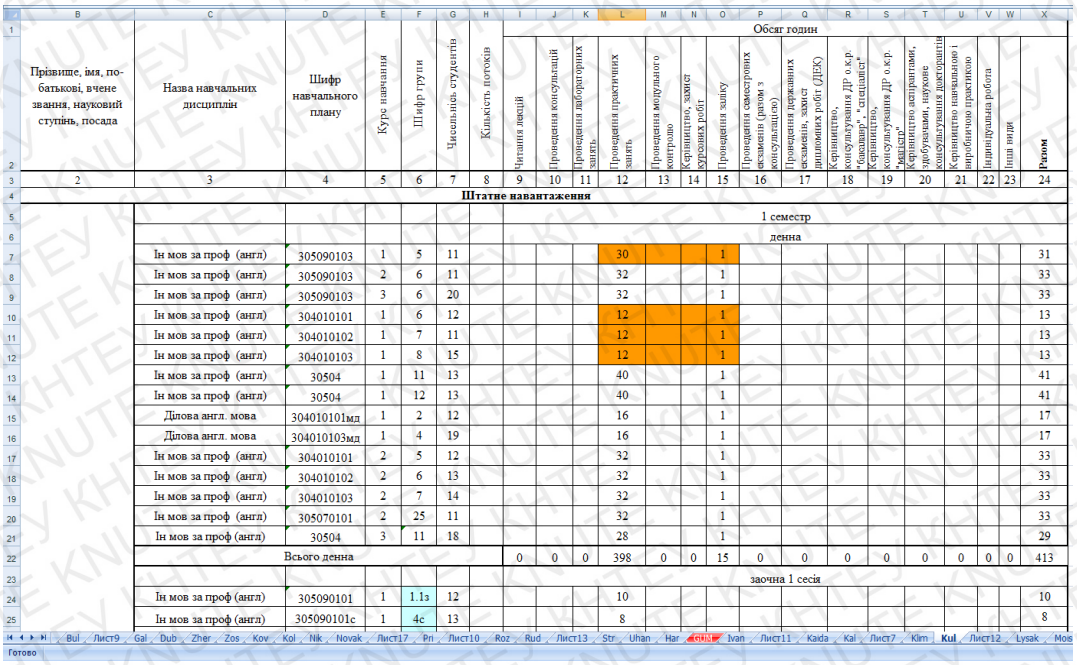


Рис. 3.3.10. Звітування

Після того як користувач розподілив навантаження на всіх викладіч кафедри, він має створити табеля на кожного викладача, та окремо на всю кафедру. У деяких випадках виклад не буде згоден з розподіленням по різним причинам. В такому разі табель не підписується, а виконується перерозподіл навантаження. Табель формується ще раз і отримується викладачем.

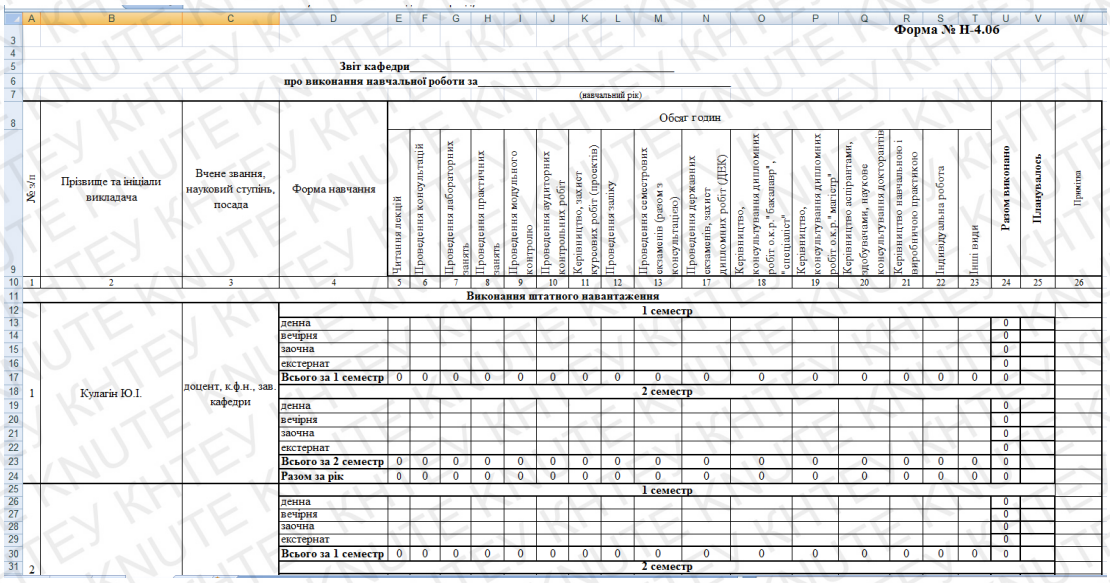
Табеля формуються на основі зібранної інформації з бази даних та мають наступний вигляд (рисунок 3.3.11, рисунок 3.3.12).

Зберігаються у файлі в форматі Excel.



Прізвище, ім'я, по-батькові, вчене звання, науковий ступінь, посада	Назва навчальних дисциплін	Шифр навчального плану	Курс навчання	Шифр групи	Чисельність студентів	Кількість потоків	Обсяг годин																
							Читання лекцій	Проведення консультацій	Проведення лабораторних занять	Проведення практичних занять	Проведення модульного контролю	Керівництво, захист курсових робіт	Проведення заліку	Проведення семестрових екзаменів (разом з консультацією)	Проведення державних екзаменів, захист дипломних робіт (ДЕК)	Керівництво, консультування дипломних робіт о.к.р. "бакалавр", "спеціаліст"	Керівництво, консультування ДР о.к.р. "магістр"	Керівництво аспірантами, здобувачами, наукове консультування докторантів	Керівництво навчальною і виробничою практикою	Індивідуальна робота	Інші види	Разом
2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	
Штатне навантаження																							
							1 семестр																
							денна																
	Ін мов за проф (англ)	305090103	1	5	11					30			1									31	
	Ін мов за проф (англ)	305090103	2	6	11					32			1									33	
	Ін мов за проф (англ)	305090103	3	6	20					32			1									33	
	Ін мов за проф (англ)	304010101	1	6	12					12			1									13	
	Ін мов за проф (англ)	304010102	1	7	11					12			1									13	
	Ін мов за проф (англ)	304010103	1	8	15					12			1									13	
	Ін мов за проф (англ)	30504	1	11	13					40			1									41	
	Ін мов за проф (англ)	30504	1	12	13					40			1									41	
	Ділова англ. мова	304010101мд	1	2	12					16			1									17	
	Ділова англ. мова	304010103мд	1	4	19					16			1									17	
	Ін мов за проф (англ)	304010101	2	5	12					32			1									33	
	Ін мов за проф (англ)	304010102	2	6	13					32			1									33	
	Ін мов за проф (англ)	304010103	2	7	14					32			1									33	
	Ін мов за проф (англ)	305070101	2	25	11					32			1									33	
	Ін мов за проф (англ)	30504	3	11	18					28			1									29	
	Всього денна						0	0	0	398	0	0	15	0	0	0	0	0	0	0	0	413	
							заочна 1 сесія																
	Ін мов за проф (англ)	305090101	1	1.1з	12					10												10	
	Ін мов за проф (англ)	305090101с	1	4с	13					8												8	

Рис. 3.3.11. Табель викладача



Форма № Н-4.06

Звіт кафедри _____

про виконання навчальної роботи за _____

(навчальний рік)

№з/п	Прізвище та ініціали викладача	Вчене звання, науковий ступінь, посада	Форма навчання	Обсяг годин: Читання лекцій	Проведення консультацій	Проведення лабораторних занять	Проведення практичних занять	Проведення модульного контролю	Проведення аудиторних контрольних робіт	Керівництво, захист курсових робіт (проекта)	Проведення заліку	Проведення семестрових екзаменів (разом з консультацією)	Проведення державних екзаменів, захист дипломних робіт (ДЕК)	Керівництво, консультування дипломних робіт о.к.р."бакалавр", "спеціаліст"	Керівництво, консультування дипломних робіт о.к.р."магістр"	Керівництво аспірантами, здобувачами, наукове консультування докторантів	Керівництво навчальною і виробничою практикою	Індивідуальна робота	Інші види	Разом виконано	Планувалось	Примітки
1	2	3	4	5	6	7	8	9	10	11	12	13	17	18	19	20	21	22	23	24	25	26
Виконання штатного навантаження																						
			1 семестр																			
			денна																	0		
			вечірня																	0		
			заочна																	0		
			екстернат																	0		
			Всього за 1 семестр	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
1	Кулагін Ю.І.	доцент, к.ф.н., зав. кафедри	**2 семестр**																			
			денна																	0		
			вечірня																	0		
			заочна																	0		
			екстернат																	0		
			Всього за 2 семестр	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
			Разом за рік	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
			1 семестр																			
			денна																	0		
			вечірня																	0		
			заочна																	0		
			екстернат																	0		
			Всього за 1 семестр	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
2			**2 семестр**																			

Рис. 3.3.12. Звіт кафедри

В розділі "Допомога" розміщений пункт "Справка", створена для інструктування користувача. В ній описується робота з програмою, дій з файлами та даними (рисунок 3.3.13).

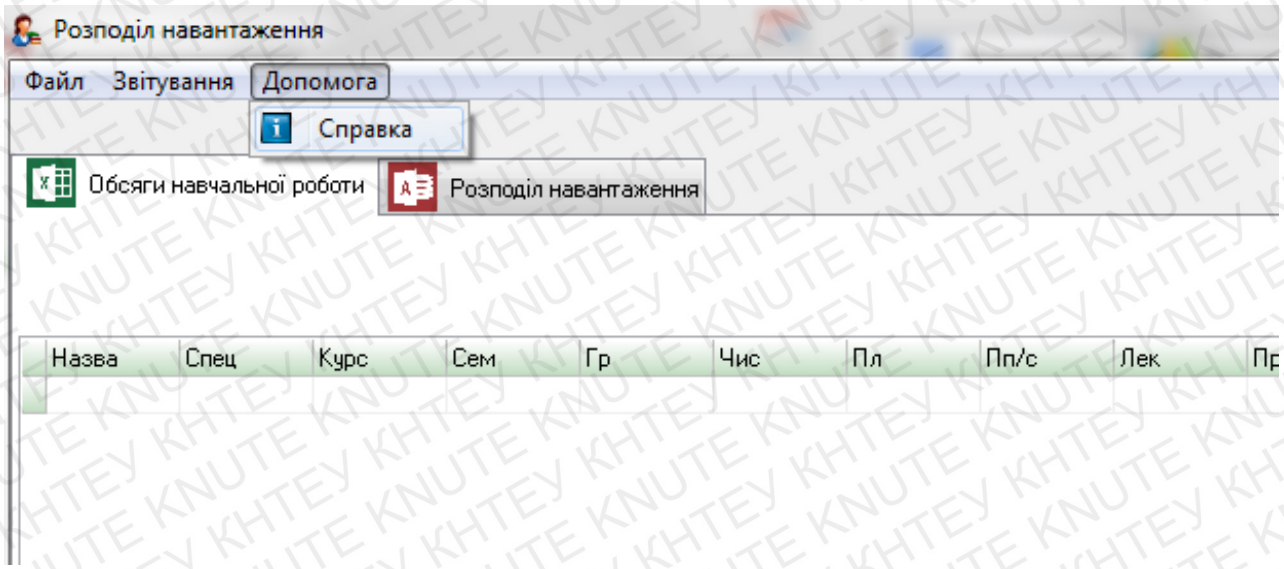


Рис. 3.3.13. Справка

ВИСНОВОК

З метою автоматизації процесу формування розподілу навантаження дисциплін, розроблено моделі автоматизованої системи документообігу та методику автоматизованого розподілу навантаження, яка враховує специфіку організації навчального процесу. Запропоновані моделі автоматизованої системи реалізовані в окремому модулі, який забезпечує процес розподілу кафедрального навантаження та оптимізує процес розподілу шляхом використання розроблених засобів автоматизації уніфікації рутинних процесів опрацювання однотипної інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Симбірська Л.М., Клітна І.В. Комп'ютерна система планування навчальної роботи ВНЗ: Збірник наукових праць // Вісник ХГАДТУ. - Харків. - 2002.- №17. - С.5 - 7.
2. Гаврилець Е.З., Медведєва О.А. Автоматизована система формування навчальних планів і розподілу навчальної навантаження викладачів кафедри ВНЗ // Сучасні наукомісткі технології. - 2007. - № 2. - С. 40-41;
3. Смолянов А.Г. Управление кафедрой: автоматизированный расчёт учебной нагрузки // Международный журнал «Символ науки». 2015. № 10.
4. Каюгина С.М. Автоматизированная система расчёта и распределения учебной нагрузки преподавателей кафедры вуза на базе платформы «1С» // Современные научные исследования и инновации. 2015. № 11.
5. Тандура Г. Підвищення ефективності управлінської діяльності шляхом її автоматизації / Г. Тандура // Управління школою. – 2007. – № 29. – С. 25-31.
6. Митчелл К. Керман Программирование и отладка в Delphi: Учебный курс: Киев, 2003. – 329 с.
7. Ломоносов О.В. Методичні положення управління чисельністю науково-педагогічних співробітників вищих навчальних закладів/ О.В. Ломоносов // Науково-методичний журнал. – Вип. 7. Економічні науки. – Миколаїв: Вид-во ім. Петра Могили, 2010. – 56-60 с.
8. Васильєва Т. Оптимальний розподіл навантаження на викладач [Електронний ресурс] / Т. Васильєва. – Режим доступу: http://www.rusnauka.com/30_NIEK_2011...
9. Васильєв В. Механізм розподілу штатів між кафедрами університету [Електронний ресурс] / В. Васильєв. – Режим доступу: http://www.ict.edu.ru/vconf...
10. Ломоносов О.В. Залежність між основними трудовими показниками вищих навчальних закладів III-IV рівнів акредитації / О.В. Ломоносов

// Науково-методичний журнал. – Вип. 86. Економічні науки. – Миколаїв: Вид-во ім. Петра Могили, 2008. – 117-124 с.

11. Концепція розподілу штатів професорсько-викладацького складу // Науково-методичний журнал. – Л.: Національний університет «Львівська політехніка». – 124-127 с.

12. Євдокимов О.В. Система розподілу навчального навантаження університету / О.В. Євдокимов // Сучасна інформаційна Україна: інформатика, економіка, філософія: матеріали доповідей конференції, 26 квітня 2012 року, Донецьк, 2012. – 316 с.

13. Болюбаш Я. Я. Організація навчального процесу у вищих закладах освіти : навч. посібник для слухачів закладів підвищення кваліфікації системи вищої освіти / Я. Я.Болюбаш. – К. : ВВП «КОМПАС», 1997. – 64 с.

14. Мацяшек Л. А. Анализ и проектирование информационных систем с помощью UML 2.0 / Л. А. Мацяшек. – М. : Вильямс, 2008. – 816 с.

15. Моисеев А.С. Object Pascal – М.: Москва, 2000. – 334 с.

16. Немнюгин С.А. Программирование – М.: Питер, 2000.-426 с.

17. Фаронов В.В. Delphi 6: Учебный курс. – СПб.: Питер, 2002. – 421 с.

18. Архангельский А.Я. Object Pascal в Delphi. – СПб.: Бином, 2002. – 265 с.

19. Галисеев Г.В. Программирование в среде Delphi 7. Самоучитель. – М.:Издательский дом «Вильямс», 2003. – 274 с.

ДОДАТОК

```
unit MianUnit;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs, Grids, DBGrids, StdCtrls, Mask, DBCtrls, DB, ADODB, ComObj,
 ToolWin, ComCtrls, ExtCtrls, Vcl.Touch.Keyboard;
type
 TMainForm = class(TForm)
  StatusBar1: TStatusBar;
  ToolBar1: TToolBar;
  PageControl1: TPageControl;
  Splitter1: TSplitter;
  PageControl2: TPageControl;
  TabSheet1: TTabSheet;
  TabSheet2: TTabSheet;
  TabSheet3: TTabSheet;
  TabSheet4: TTabSheet;
  DBGridExcel: TDBGrid;
  ToolBar2: TToolBar;
  ToolButton1: TToolButton;
  ToolButton2: TToolButton;
  DBGridAccessCal: TDBGrid;
  DBNavigator1: TDBNavigator;
  DBNavigator2: TDBNavigator;
  DBGrid2: TDBGrid;
  DBLookupComboBox1: TDBLookupComboBox;
  ADOQueryActiveCalculate: TADOQuery;
  DataSourceActiveCalculate: TDataSource;
  GroupBox1: TGroupBox;
  Label1: TLabel;
  RadioGroup1: TRadioGroup;
  DBLookupComboBox2: TDBLookupComboBox;
```

```
    ADOQuery1: TADOQuery;
    DataSource1: TDataSource;
    DBGrid3: TDBGrid;
    Label2: TLabel;
    GroupBox2: TGroupBox;
    EditFilterFIO: TLabeledEdit;
    DBGrid1: TDBGrid;
    ADOConnection1: TADOConnection;
    btn1: TButton;
    RadioGroup2: TRadioGroup;
    Label3: TLabel;
    ADOQuery2: TADOQuery;
    ADOTable1: TADOTable;
    DataSource2: TDataSource;
    procedure FormCreate(Sender: TObject);
    procedure DBGridAccessCalDrawColumnCell(Sender: TObject;
      const Rect: TRect; DataCol: Integer; Column: TColumn;
      State: TGridDrawState);
    procedure ToolButton1Click(Sender: TObject);
    procedure DBLookupComboBox1Click(Sender: TObject);
    procedure EditFilterFIOChange(Sender: TObject);
    procedure DBGrid1TitleClick(Column: TColumn);
    procedure DBGrid2TitleClick(Column: TColumn);
    procedure ToolButton2Click(Sender: TObject);
    procedure btn1Click(Sender: TObject);
   // procedure btn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  MainForm: TMainForm;
  function DBConnection(const ADBName: string): Boolean;
```

```
 function ConvertExcelTOAccess (DataArray : array of string): Boolean;
implementation
uses DataBaseModuleUnit, DataModulMainForm, ExcelConnectionDataModul;
{$R *.dfm}
function ConvertExcelTOAccess (DataArray : array of string): Boolean;
var
 i: Integer;
begin
 with DataBaseModule.ADOTableCalculate do
  begin
   //Добавить новую запись в таблицу Нагрузка
   Insert;
   for i:=1 to 24 do
    begin
     ShowMessage(IntToStr(i)+' '+DataArray[i-1]);
     Fields[i].Value:=DataArray[i-1];
    end;

   //Сохранить новую запись
   Post;
  end;
 Result:=True;
end;

function DBConnection(const ADBName: string): Boolean;
var
 {ConnString,} DatabaseName: string;
begin
 //В конце доправить
 DatabaseName := IncludeTrailingPathDelimiter(ExtractFilePath(ParamStr(0))) + ADBName;
 ShowMessage(DatabaseName);
 DataBaseModule.ADOConAccess.ConnectionString:=      'Provider=Microsoft.Jet.OLEDB.4.0;Data
            Source='+DatabaseName+';Persist Security Info=False;';
 DataBaseModule.ADOTableCalculate.Active:=True;
```

```
  DataBaseModule.ADOTableDistribution.Active:=True;
  DataBaseModule.ADOQueryTeachersWithSUM.Active:=True;
  DataBaseModule.ADOQueryCalculate.Active:=True;
  Result := DataBaseModule.ADOConAccess.Connected;
end;
procedure TMainForm.FormCreate(Sender: TObject);
var
ProgressBar1 : TProgressBar;
begin
  //Создание прогресс бара в Статусбаре
  ProgressBar1:=TProgressBar.Create(MainForm);
  with ProgressBar1 do
   begin
    Parent := StatusBar1;
    Smooth:=True;
    Position:=20;
    Top := 2;
    Left := 300;
    Height := StatusBar1.Height - Top;
    Width := StatusBar1.Panels[0].Width - Left;
   end;
end;

//Процедура перерисовки boolean в Checkbox====================================
procedure DrawGridCheckBox(Canvas: TCanvas; Rect: TRect; Checked: boolean);
var
  DrawFlags: Integer;
begin
  Canvas.TextRect(Rect, Rect.Left + 1, Rect.Top + 1, ' ');
  DrawFrameControl(Canvas.Handle,  Rect,  DFC_BUTTON,  DFCS_BUTTONPUSH  or
            DFCS_ADJUSTRECT);
  DrawFlags := DFCS_BUTTONCHECK or DFCS_ADJUSTRECT;// DFCS_BUTTONCHECK
  if Checked then
   DrawFlags := DrawFlags or DFCS_CHECKED;
```

```
  DrawFrameControl(Canvas.Handle, Rect, DFC_BUTTON, DrawFlags);
end;
//============================================================================
procedure TMainForm.DBGridAccessCalDrawColumnCell(Sender: TObject;
 const Rect: TRect; DataCol: Integer; Column: TColumn;
 State: TGridDrawState);
 var
 CheckLekc, CheckPract : Boolean;
begin
  {if (DataBaseModule.ADOTableCalculate.FieldByName('Чтение_лекций').Value <> null) and
 (DataBaseModule.ADOTableCalculate.FieldByName('Чтение_практики').Value  <>  null)  then
          begin
  CheckLekc:=DataBaseModule.ADOTableCalculate.FieldByName('Чтение_лекций').Value;
  CheckPract:=DataBaseModule.ADOTableCalculate.FieldByName('Чтение_практики').Value;}
  {if (DataBaseModule.ADOQueryCalculate.FieldByName('Чтение_лекций').Value <> null) and
  (DataBaseModule.ADOQueryCalculate.FieldByName('Чтение_практики').Value  <>  null)  then
          begin
  CheckLekc:=DataBaseModule.ADOQueryCalculate.FieldByName('Чтение_лекций').Value;
  CheckPract:=DataBaseModule.ADOQueryCalculate.FieldByName('Чтение_практики').Value;
  end else begin
  CheckLekc:=False;
  CheckPract:=False;
  end;
  //Если в поле "Чтение лекции" стоит пометка
  if (CheckLekc) and (CheckPract=False) then begin
  with  DBGridAccessCal.Canvas do begin
  Brush.Color:=clOlive;
  Font.Color:=clWhite;
  FillRect(Rect);
  TextOut(Rect.Left+2,Rect.Top+2,Column.Field.Text);
  end;
  //Если в поле "Чтение практики" стоит пометка
  end else if (CheckPract) and (CheckLekc=False) then begin
  with  DBGridAccessCal.Canvas do begin
```

```
    Brush.Color:=clBlue;
    Font.Color:=clWhite;
    FillRect(Rect);
    TextOut(Rect.Left+2,Rect.Top+2,Column.Field.Text);
    end;
    //Если в поле "Чтение лекции" и "Чтение практики"  стоит пометка
    end else if (CheckLekc) and (CheckPract) then begin
     with  DBGridAccessCal.Canvas do begin
    Brush.Color:=clGreen;
    Font.Color:=clWhite;
    FillRect(Rect);
    TextOut(Rect.Left+2,Rect.Top+2,Column.Field.Text);
    end;
    end;
    if (Column.FieldName = 'Чтение_лекций') or (Column.FieldName = 'Чтение_практики')  then
    if Column.Field.AsBoolean then
      DrawGridCheckBox(DBGridAccessCal.Canvas, Rect, true)
    else
      DrawGridCheckBox(DBGridAccessCal.Canvas, Rect, false);}

end;
procedure TMainForm.ToolButton1Click(Sender: TObject);
var
i:integer;
begin
  try
    if DataModuleMainForm.OpenDialog1.Execute then
      begin
        DataExcelModul.ADOConExcel.Connected:=False;
        DataExcelModul.ADOConExcel.ConnectionString:='Provider=MSDASQL.1;Persist Security
            Info=True;Extended Properties="DSN=Excel
            Files;DBQ='+DataModuleMainForm.OpenDialog1.FileName+';DefaultDir=C:\Users\Ma3
            str0o\Desktop\New Load Calculation\DB
            LocadCalculation;DriverId=1046;MaxBufferSize=2048;PageTimeout=5;"';
```

```
      DataExcelModul.ADOConExcel.Connected:=True;
      DataExcelModul.ADOQueryExcel.Active:=False;
      DataExcelModul.ADOQueryExcel.SQL.Clear;
      DataExcelModul.ADOQueryExcel.SQL.Add('SELECT * FROM [Лист1$]');
      DataExcelModul.ADOQueryExcel.Active:=True;
    end;
  Except
   on e:Exception do
  end;
  //Первая колонка таблицы Ексель = 260
  DBGridExcel.Columns[0].Width:=260;
  //Ширина колонок для таблицы Ексель = 45
  for i:=1  to 31 do
   begin
    DBGridExcel.Columns[i].Width:=45;
   end;
end;

procedure TMainForm.DBLookupComboBox1Click(Sender: TObject);
var
 IdTeacher : String;
begin
 IdTeacher:=IntToStr(DBLookupComboBox1.KeyValue);
end;
procedure TMainForm.EditFilterFIOChange(Sender: TObject);
begin
 //Если в поле фильтрации что то есть, тогда начать фильтровать
 If EditFilterFIO.Text<>'' then begin
  DataBaseModule.ADOQueryTeachersWithSUM.Filtered:=False;
  //Поиск по любым совпадениям в поле ФИО
  DataBaseModule.ADOQueryTeachersWithSUM.Filter:='[ФИО]    LIKE    '+    ''''    +    '%'    +
            EditFilterFIO.Text + '%' + '''';
  DataBaseModule.ADOQueryTeachersWithSUM.Filtered:=True;
 end else
```

```
  DataBaseModule.ADOQueryTeachersWithSUM.Filtered:=False;
end;
procedure TMainForm.DBGrid1TitleClick(Column: TColumn);
begin
 ADOQueryActiveCalculate.Sort:=Column.FieldName;
end;


procedure TMainForm.DBGrid2TitleClick(Column: TColumn);
begin
 DataBaseModule.ADOQueryTeachersWithSUM.Sort:=Column.FieldName;
end;
procedure TMainForm.ToolButton2Click(Sender: TObject);
var
 DataArray: array[0..31] of string;
 i : Integer;
begin
//Установка метки на первую запись таблицы Excel
 DBGridExcel.DataSource.DataSet.First;
 //Пока Таблица не закончилась, перебирать ячейки
 While  not DBGridExcel.DataSource.DataSet.Eof do begin
 //Считывание данных из таблицы в массив (по строчно)
  for i:=0 to length(DataArray)-1 do begin
   DataArray[i]:=DBGridExcel.DataSource.DataSet.Fields[i].AsString;
   if DataArray[i]='' then begin
    DataArray[i]:='0';
   end else if DataArray[i]='I' then begin
    DataArray[i]:='1';
   end else if DataArray[i]='3'then
    DataArray[i]:='3';
  end;
  //Тут написать функцию отправки даных из массива в Аксесс
  ShowMessage (BoolToStr(ConvertExcelTOAccess(DataArray)));
  //Переход на следующую строку таблицы
```

```
  DBGridExcel.DataSource.DataSet.Next;
 end;
end;
{procedure TMainForm.btn1Click(Sender: TObject);
var
 c: shortint;
 s: string;
 ABitmap: TBitmap;
begin
 s := 'A';
 ABitmap:=TBitmap.Create;
 ABitmap.LoadFromFile(ExtractFilePath(Application.ExeName)+'\Image\trash-icon.bmp');
 with DBNavigator1 do
  for c := 0 to ControlCount - 1 do
   if Controls[c] is TNavButton then
    with TNavButton(Controls[c]) do
    begin
     Glyph := ABitmap;
     Caption := '1';
     Inc(s[1]);
    end;
end;}
procedure TMainForm.btn1Click(Sender: TObject);
var
 LessonType : String;
begin
// DBLookupComboBox2.KeyValue - id_препода (String)
// RadioGroup1.ItemIndex (Integer)
 case RadioGroup1.ItemIndex of
  0 : LessonType:='лекция';
  1 : LessonType:='практика';
  2 : LessonType:='лаба';
 end;
 ADOQuery2.Parameters.ParamByName('lestype').Value := LessonType;
```

```
ADOQuery2.SQL.Text:='Select COUNT(id_нагрузки) AS temp From Распределение WHERE
        id_нагрузки=177 AND тип_занятия=:lestype GROUP BY id_нагрузки';
ADOQuery2.Open;
if ADOQuery2.FieldValues['temp']<>null then begin
 ShowMessage(Inttostr(ADOQuery2.FieldValues['temp']));
end else
 ShowMessage('0');
end;
end.
```