МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет магістерської та
аспірантської підготовки
Кафедра інформаційних
технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Інтеграція СБД «Навантаження викладачів та розклад занять»

Виконав студент 2 курсу групи МК-61
спеціальності 8.05010101 Інформаційні
управляючі системи та технології,
Павленко Ігор Володимирович

Керівник к.ф.-м.н., доцент,
Козловська Валентина Петрівна

Консультант

Рецензент к.т.н., доцент,
Худенко Надія Петрівна

Одеса 2017

АНОТАЦІЯ

Тема магістерської роботи: «Інтеграція СБД «Навантаження викладачів та розклад занять»». Магістерська робота є частиною комплексної магістерської роботи «Інтеграція у єдину інформаційну систему «Університет» прикладних СБД», яка призначена створенню інформаційної системи «Університет» на основі інтегрування декількох прикладних систем баз даних, що описують різні аспекти навчального процесу в університеті.

Актуальність теми: визначається необхідністю розробки інформаційної системи, що охоплює всі аспекти навчального процесу в університеті.

Метою і задачею дослідження є створення інформаційної системи «Університет» на платформі СКБД MS SQL Server для вирішення питань автоматизації всіх стадій навчального процесу – від розробки навчальних планів до складання розкладу заняття та моніторингу поточної успішності студентів.

Об’єктом дослідження є навчальний процес вищого навчального закладу.

Предметом дослідження є концептуальне моделювання навчального процесу вищого навчального закладу.

Методи дослідження. Використовуються методи теорії відношень та реляційної алгебри.

Результатом даної роботи є розробка інформаційної системи «Університет» з програмним застосуванням «АРМ працівника кафедри».

Науковою новизною роботи є інтеграція в єдину базу даних прикладних систем баз даних, що описують різні аспекти навчального процесу в університеті.

Теоретичним значенням є розробка концептуальної моделі навчального процесу у вищому навчальному закладі.

Практичним значенням є розробка бази даних навчального процесу ВНЗ.

Результати даної роботи можуть використовуватись у вищих навчальних закладах.

Вихідні дані: інформація про навчальний процесс ВНЗ.

Об’єм роботи складає – 76 стор., кількість рисунків – 21, кількість використаної літератури – 15.

Ключові слова: база даних, навчальний процес, навантаження викладача.

SUMMARY

Theme of master's work: "The integration DBS “The teacher’s load and schedule”". Master's work is part of a comprehensive master's thesis "The integration into a single information system "University" application DBS", which is creating an information "University" system, based on the integration of multiple application database systems that describe the various aspects of the educational process at the university.

Relevance: determined the need to develop an information system covering all aspects of the educational process at the university.

The purpose and objective of the study is to create an information system "University" on the MS SQL Server DBMS platform for addressing the automation of all stages of the learning process – from curriculum design to scheduling classes and monitoring the current students' progress

The object is the educational process of higher educational institutions.

The subject of the research is the conceptual design of the educational process of university

Research methods. The methods of the theory of relations and relational algebra.

The result of this work is the development of information system "University" and application software "Workstation of the Department’s employee".

Scientific novelty of the work is to integrate into a single database application systems database describing the various aspects of the educational process at the university.

The theoretical value is to develop a conceptual model of the educational process in higher education.

The practical significance is the development of educational process of the university database.

The results of this study can be used in higher educational institutions.

Input data: Information on the educational process of higher educational institutions.

The volume of work is – 76 p., the number of figures – 21, the number of references – 15.

Keywords: database, the learning process, the teacher’s load.

ЗМІСТ

ВСТУП

В Одеському державному екологічному університеті на протязі останніх років при виконанні дипломних проектів та магістерських робіт розроблялись інформаційні системи, що описують різні аспекти навчального процесу в університеті. Прикладом подібних інформаційних систем, або прикладних баз даних, можуть слугувати такі системи: «Електронний деканат», «Тестуюча система», «Інтегральні відомості», «Навантаження викладача», «Розклад занять в університеті».

Всі ці системи розроблялись як окремі незалежні системи баз даних. Подібний підхід є помилковим, оскільки у перерахованих базах даних міститься дуже багато спільної інформації, а саме: перелік підрозділів університету, список викладацького составу університету, перелік навчальних дисциплін, навчальні плани факультетів, тощо. Для можливості зручної роботи користувачів з перерахованими системами потрібно об’єднати їх у єдину інформаційну систему «Університет».

Метою даної комплексної магістерської роботи є інтегрування у єдину інформаційну систему «Університет» вказаних прикладних систем баз даних.

Задачею даної частини комплексної магістерської роботи є інтегрування у єдину базу даних інформаційної системи «Університет» баз даних «Навантаження викладача» та «Розклад занять», а також розробка офісного клієнтського застосування для групи користувачів «Працівник кафедри».

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

При проектуванні ІС «Університет» необхідно враховувати потоки даних, які існують в університеті при складанні розкладу занять студентів.

Протягом навчального року декани розробляють навчальні плани наступного навчального року. У навчальних планах вказується кафедра, яка читає цю дисципліну.

Після затвердження навчальних планів методичним відділом, вони розсилаються по мережі для використання їх кафедрами і навчальним відділом. Також методичний відділ розсилає список дисциплін на наступний навчальний рік і прив'язку їх до кафедр.

Кафедри розподіляють аудиторне навантаження по закріплених дисциплінах між викладачами і передають цю інформацію в навчальний відділ для складання розкладу занять на наступний семестр. Також в навчальний відділ передається інформація про допустимі аудиторії для проведення кожного заняття. Аудиторне навантаження є основою для визначення навчального навантаження викладача.

Навчальний відділ по навчальним планам факультетів та інформації, що надходить з кафедр, складає розклад занять.

Потік даних між підрозділами університету у процесі складання розкладу занять зображено на рис. 1.1.

При наявності ІС «Університет» методичний відділ після затвердження навчальних планів вводить в базу даних список дисциплін на наступний навчальний рік і прив'язку їх до кафедр.

Диспетчери факультетів вводять в базу даних навчальні плани на наступний навчальний рік, передбачуваний контингент студентів, прив'язку академічних груп до навчальних планів факультету.

За цими даними автоматично формується список занять кожної групи на розрахунковий семестр.

Список занять груп є вихідними даними для кафедр для розподілу аудиторного навантаження між викладачами.

Співробітники, відповідальні за розрахунок навантаження викладачів на кафедрах, додають в базу даних інформацію про те, в якому потоці читається кожна лекція, а також які групи діляться на підгрупи при проведенні лабораторних занять – для кожної дисципліни кафедри може окремо задаватися, чи буде ділитись група на підгрупи.

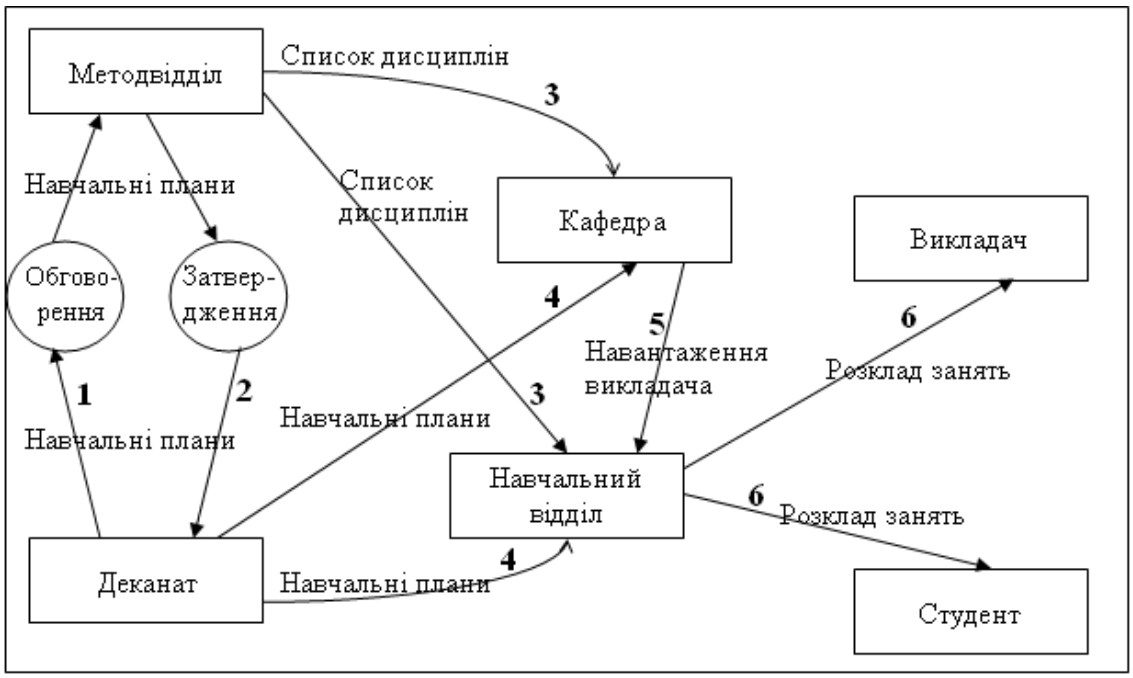


Рисунок 1.1 – Контекстна діаграма потоків даних процесу складання розкладу занять

Для кожного отриманого заняття: лекції, семінару, лабораторної роботи, – в базі даних вибирається викладач зі списку викладачів, наявного в базі даних (необов'язково тільки з викладачів даної кафедри). Також для кожного заняття на кафедрі призначається список допустимих аудиторій.

Аудиторне навантаження викладачів є основою для складання розкладу занять. Також навантаження від проведення занять в обох семестрах навчального року є основою навчального навантаження кафедри. Крім проведення занять до навчального навантаження відносяться і інші види роботи викладачів: консультації з навчальних дисциплін, проведення навчальної практики, керівництво дипломними проектами та магістерськими роботами, керівництво курсовими роботами та проектами. перевірка модульних контрольних робіт та інше.

Навчальне навантаження, що зв’язане з якою-небудь навчальною дисципліною, розраховується згідно з даними, що прописані і відповідному пункті навчального плану: кількість годин лекцій, кількість годин практичних занять, кількість годин лабораторних занять, кількість годин занять для навчальної практики.

Деякі види навчального навантаження розраховуються відповідно до даних з дисципліни в навчальному плані, але для їх розрахунку використо-

вуються додаткові дані, що визначають розмір навчального навантаження викладача на виконання заданого виду навчальної роботи. До цих видів навчального навантаження відносяться консультацій, керівництво курсовими роботами та проектами, перевірка модульних контрольних робіт та інших видів контролю самостійної роботи студентів (СРС), проведення заліку та іспиту, перевірка письмових екзаменаційних робіт, тощо.

До видів навчального навантаження, що не зв'язані з якою-небудь навчальною дисципліною відносяться керівництво дипломними проектами та магістерськими роботами. Для цих видів навантаження також використовуються додаткові дані, що визначають розмір навчального навантаження викладача на виконання заданого виду навчальної роботи.

Список видів навчальної роботи викладачів, для яких окремо визначається навчальне навантаження, повинен задаватися методичним відділом університету. В цьому списку для кожного виду додаткового навчального навантаження повинно вказуватись розмір навантаження за виконання одиниці данного виду роботи, та визначатись, вважається за одиницю виконання роботи на одного студента чи на одну академічну групу.

Працівник кафедри, відповідальний за розрахунок навантаження викладачів, розподіляє між викладачами ці додаткові види навчального навантаження: консультації з дисципліни, керівництво дипломними проектами та магістерськими роботами, керівництво курсовими роботами та проектами.

Для визначення навчального навантаження від контролю викладачем самостійної роботи студентів необхідно визначити у кожному семестрі для кожної навчальної дисципліни склад та кількість видів СРС. Потім орозраховане навчальне навантаження також потрідно розподілити між викладачами.

При використанні для розрахунку навчального навантаження ІС «Університет» автоматично розраховується навантаження для кожної навчальної дисципліни для кожною групи – згідно контингенту студентів та навчальних планів, до яких закріплюється група на розрахунковий навчальний рік. Також автоматично розраховується навантаження на керівництво дипломними проектами або магістерськими роботами для кожної групи, у навчальних планах яких є виконання дипломного проекту або магістерської роботи.

У навчальних планах для кожної навчальної дисципліни вказується кафедра, яка повинна вести цю дисципліну. Таким чином навчальне навантаження, обумовлене проведенням всіх видів навчальних робіт по всіх дисциплінах кафедри, також розраховується автоматично, крім навантаження, що обумовлено контролюючими заходами СРС. Для розрахунку останнього виду

навчального навантаження потрібно задати для кожної дисципліни види контролю СРС.

Навчальне навантаження кафедри автоматично перераховується, коли для дисципліни вказується, що лекції проводяться не для кожної групи окремо, а на потоці з декількох груп. Також автоматично перераховується навчальне навантаження кафедри, коли для деяких дисциплін вказується, що лабораторні заняття проводяться з розподілом академічної групи на підгрупи.

Для розрахунку навчального навантаження від керівництва дипломними проектами або магістерськими роботами потрібно вказати, студентами яких груп будуть керувати викладачі кафедри, та вказати кількість студентів, якщо вказане, що керівництво для цієї групи можуть здійснювати викладачі не однієї кафедри.

Для автоматичного складання розкладу занять від кафедри додатково потрібно надати інформацію про допустимі аудиторії для кожного заняття.

До вихідних даних для складання розкладу занять також відносяться вимоги та побажання викладачів до часу проведення занять – дні тижня та навчальні пари; наприклад, для викладачів, які зайняті іншими видами діяльності у деякі дні тижня.

При наявності у складі ІС «Університет» програми автоматичного складання розкладу занять диспетчер навчального відділу використовує її для складання розклад занять на наступний семестр. Якщо такої програми немає, співробітник навчального відділу за допомогою програми «АРМ диспетчера навчального відділу» складає розклад на комп'ютері.

Якщо у складі ІС «Університет» є обидві програми: програма автоматичного складання розкладу занять та програма «АРМ диспетчера навчального відділу», – бажано скласти першу версію розкладу занять за допомогою програми автоматичного складання розкладу занять, а потім за бажанням відкоригувати розклад за допомогою програми «АРМ диспетчера навчального відділу».

Після складання розкладу і установки для нього позначки про готовність користувачі-викладачі отримують можливість перегляду на сайтах кафедр отриманого розкладу для можливості завчасного внесення в нього змін.

Після внесення до розкладу занять необхідних змін викладачі, студенти і гості отримують можливість перегляду розкладу занять на сайтах факультетів та кафедр перед початком занять (і протягом семестру).

У минулі роки були розроблені окремі бази даних «Навантаження викладача» і «Розклад занять». Однак основна частина інформації з бази даних

«Навантаження викладача» використовується в базі даних «Розклад занять» в якості вихідних даних для складання розкладу занять студентів. Тому необхідно інтегрувати ці бази даних в єдину базу даних. Більш того, ці бази даних істотно перетинаються з базами даних «Електронний деканат», «Інтегральні відомості» і «Тестуюча система»: у всіх цих базах даних міститься інформація про кафедри, факультети, академічні групи, навчальні плани, викладачів. Тому доцільно об'єднати всю інформацію в одну базу даних.

2 СИСТЕМНИЙ АНАЛІЗ І КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Після збору зовнішніх користувальницьких уявлень потрібно виконати проектування бази даних для програми, що розробляється.

2.1 Етапи проектування бази даних

Процес проектування бази даних складається з трьох основних етапів: концептуальне, логічне і фізичне проектування [4; 13].

Завданням концептуального проектування є отримання концептуальної моделі інформаційної системи, що не залежить від будь-яких фізичних аспектів представлення інформації. Створена концептуальна модель є джерелом інформації для етапу логічного проектування бази даних.

Основні етапи концептуального проектування:

- інтеграція зовнішніх користувальницьких уявлень;
- визначення типів сутностей;
- визначення типів зв'язків;
- визначення атрибутів і зв'язування їх з типами сутностей і зв'язків;
- визначення доменів атрибутів;
- визначення атрибутів, що є потенційними і первинними ключами;
- перевірка моделі на відсутність надмірності;
- перевірка відповідності локальної концептуальної моделі конкретним транзакціях користувачів;
- обговорення локальних концептуальних моделей даних з кінцевими користувачами.

Завданням логічного проектування бази даних є створення логічної моделі на основі обраної моделі даних, але без урахування конкретної СУБД, яка буде використовуватися, та інших фізичних аспектів реалізації інформаційної системи. На етапі логічного проектування отримана концептуальна модель уточнюється і перетвориться для відповідності структурам даних і зв'язків між ними, притаманних обраної моделі даних: реляційної, об'єктно-орієнтованої, об'єктно-реляційної тощо.

Логічне проектування залежить від обраної моделі даних. Для реляційної моделі можна виділити наступні етапи:

- створення і перевірка локальної логічної моделі на основі зовнішніх уявлень кожної групи користувачів;
- усунення особливостей локальної логічної моделі, несумісних з реляційною моделлю, наприклад, усунення зв'язків типу “багато-до-багатьох”;
- визначення набору відносин, виходячи зі структури локальної логічної моделі даних;
- перевірка відносин за допомогою правил нормалізації;
- перевірка відповідності відносин вимогам користувальницьких транзакцій;
- визначення вимог підтримки цілісності даних;
- створення і перевірка глобальної логічної моделі.

На останньому етапі фізичного проектування обирається конкретна СУБД і на основі логічної моделі створюється фізична схема бази даних. Основними етапами фізичного проектування для реляційної СУБД являється:

- перенесення глобальної логічної моделі в середу конкретної обраної СУБД;
- проектування базових відносин у середовищі обраної СУБД;
- проектування похідних відносин;
- реалізація обмежень цілісності предметної області;;
- визначення індексів;
- розробка користувальницьких уявлень.

2.2 Концептуальне проектування бази даних

2.2.1 Визначення основних сутностей

З опису предметної області можна виділити основні групи користувачів ІС «Університет» і їх транзакції.

Група користувачів «Диспетчер деканату»:

1) Отримати затверджений методичним відділом список навчальних дисциплін на наступний навчальний рік.
2) Відновити або визначити навчальні плани на наступний навчальний рік.
3) Затвердити в методичному відділі навчальні плани на наступний навчальний рік.

4) Відновити контингент студентів на наступний навчальний рік.
5) Відновити прикріплення академічних груп до навчальних планів.
6) Отримати готовий розклад занять.

Група користувачів «Працівник кафедри»:

1) Отримати дані за навчальними планами, контингентом студентів і прикріпленням груп до навчальних планів.
2) Отримати список дисциплін кафедри на наступний навчальний рік.
3) Визначити списки дисциплін, для яких лекції читаються на потоках і назви потоків.
4) Визначити списки дисциплін і груп, для яких лабораторні заняття або навчальні практики проводяться з розбивкою на підгрупи.
5) Прикріпити викладача до кожного заняття з дисциплін кафедри.
6) Визначити для кожної дисципліни кафедри список видів контролюючих заходів СРС.
7) Визначити академічні групи, в яких керівництво дипломними проектами або магістерськими роботами будуть вести викладачі кафедри, та кількість студентів з цих груп (якщо кафедра керує не всіма студентами групи).
8) Відновити дані про розподіл неаудиторного навчального навантаження викладачів.
9) Оновити дані про допустимі аудиторії для занять.
10) Визначити вимого та побажання викладачів до розкладу занять на наступний семестр.
11) Отримати розклад занять для викладачів кафедри на поточний та / або на наступний семестр.

Група користувачів «Диспетчер навчального відділу»:

1) Отримати дані, необхідні для складання розкладу занять, від деканатів факультетів і кафедр.
2) Скласти розклад занять.

Група користувачів «Викладач»:

1) Отримати навчальне навантаження на наступний навчальний рік
2) Отримати розклад занять на наступний семестр.

Група користувачів «Студент» – отримати розклад занять.

Таким чином, є такі основні базові типи сутностей в базі даних: Факультет, Кафедра, Навчальний_план, Група, Викладач, Предмет (навчальна дисципліна), Вид_навантаження, Вид_занять, Заняття.

Факультет має декілька академічних груп та декілька навчальних планів – між типами сутностей Факультет та Група існує тип зв’язку «один до багатьох»; між типами сутностей також Факультет та Навчальний_план існує тип зв’язку «один до багатьох».

Протягом одного навчального року кожна академічна група навчається за деяким одним навчальним планом, але декілька груп можуть іноді навчатись за одним навчальним планом – особливо це стосується перших курсів, коли вивчається небагато спеціалізованих дисциплін. Таким чином між типом сутностей Навчальний_план і типом сутностей Група існує тип зв’язку «один до багатьох».

Деякі навчальні дисципліни можуть зустрічатись у декількох навчальних планах, наприклад, загальноосвітні дисципліни. У різних навчальних планах для таких дисциплін може передбачатись різне аудиторне навантаження, тобто різна кількість занять різного виду. Тому необхідно у схему бази даних додати похідний тип сутності Пункт_плану (пункт навчального плану), відповідно до якого і потрібно визначати заняття з деякої дисципліни для деякої групи.

Кожен навчальний план містить декілька пунктів, кожен пункт плану стосується лише однієї дисципліни, але будь-яка дисципліна (предмет) може зустрічатись у декількох пунктах навчальних планів, як у різних навчальних планах, так і у тому самому навчальному плані у обох семестрах. Таким чином між типами сутностей Навчальний_план та Пункт_плану існує тип зв’язку «один до багатьох»; так саме, як і між типами сутностей Предмет та Пункт_плану.

З одного пункту навчального плану зазвичай виходить декілька занять, можливо різного виду: лекції, семінари, лабораторні заняття. Між типами сутностей Пункт_плану та Заняття існує тип зв’язку «один до багатьох»; так саме, як і між типами сутностей Вид_занять та Заняття.

На кожній кафедрі працює декілька викладачів, та кожен викладач, як правило, проводить декілька занять. Між типами сутностей Викладач та Заняття існує тип зв’язку «один до багатьох»; так саме, як і між типами сутностей Кафеда та Викладач.

Визначивши типи зв'язків між виявленими типами сутностей, отримаємо основну ER-діаграму бази даних (рис. 2.1).

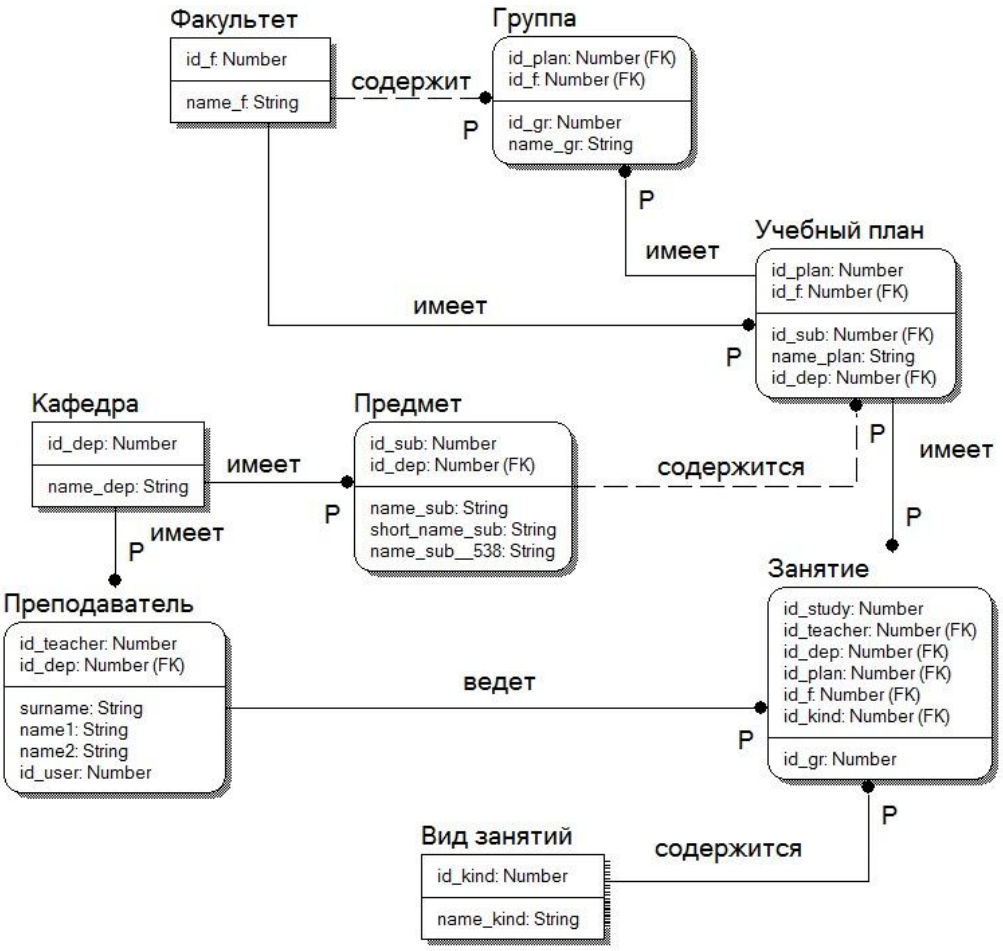


Рисунок 2.1 – Основна ER-діаграма підсхеми «Розклад занять» бази даних ІС «Університет»

Заняття повинно проводитися в якийсь аудиторії. Деякі лекції читаються на потоці з декількох груп. Для складання розкладу занять ця інформація також повинна зберігатися в базі даних. Отримуємо два нових типи сутностей: Потік і Аудиторія.

Визначивши типи зв'язків між виявленими типами сутностей, отримаємо доповнену ER-діаграму бази даних для підсхеми «Розклад занять» (рис. 2.2).

В навчальних планах для деяких дисциплін крім аудиторних занять протягом семестру, для яких складається розклад занять, передбачені інші види навчального навантаження – курсовий проект (робота), навчальна практика. Ці види навантаження не входять у схему БД «Розклад занять», але вони відносяться до об’єднаної схеми БД «Навантаження викладача» та «Розклад занять».

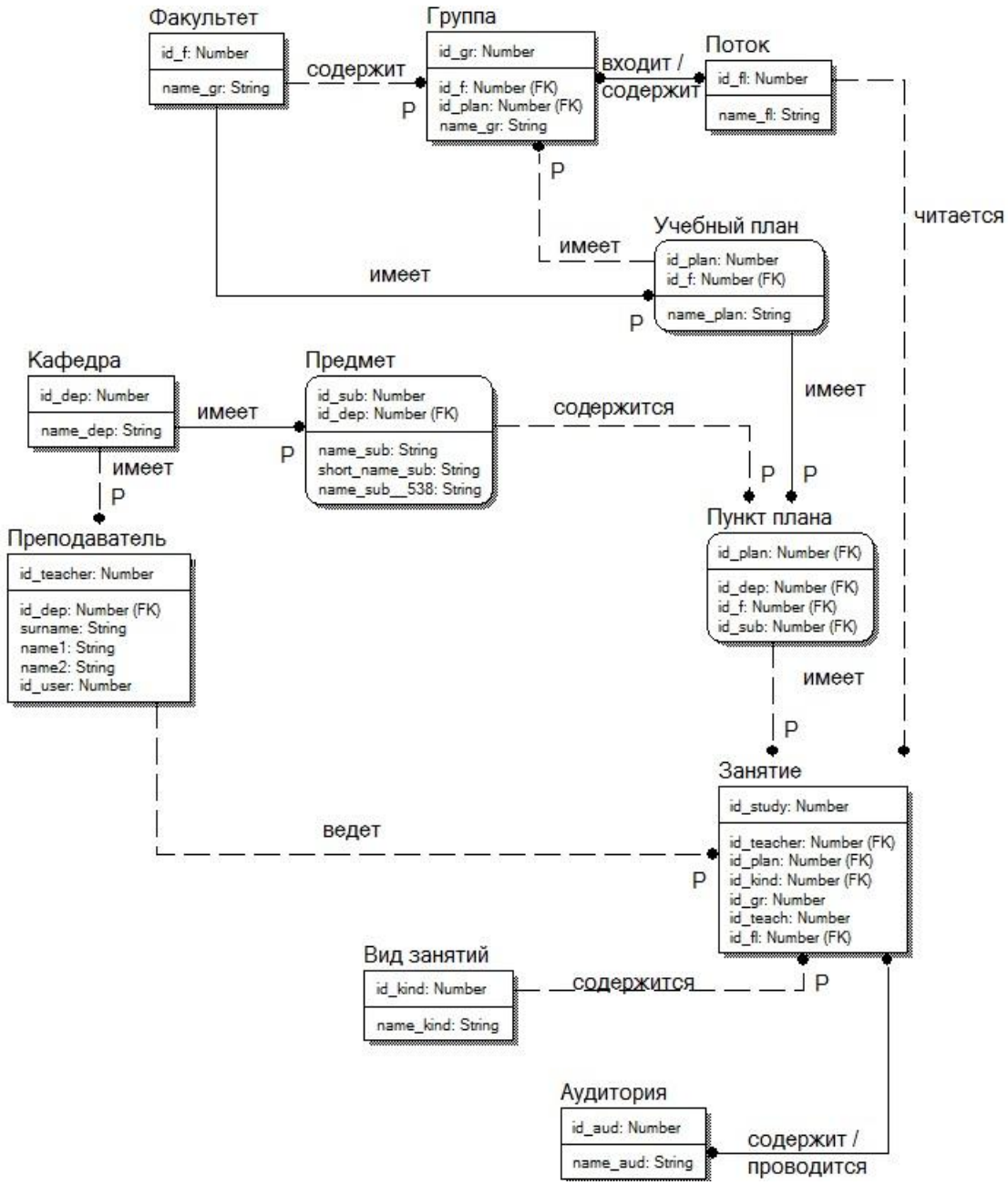


Рисунок 2.2 – Доповнена ER-діаграма підсхеми «Розклад занять» бази даних ІС «Університет»

Наявність курсового проекту або курсової роботи міститься у тому ж пункті навчального плану, що і запис про різні види занять з дисципліни. Навчальна практика з деякої дисципліни є окремим рядком навчального плану; при цьому заняття з дисципліни і навчальна практика з цієї ж дисципліни можуть проводитись у різних семестрах.

Тобто, типи зв’язку між типами сутностей Пункт_плану та Курсовик (курсова робота або курсовий проект) буде умовний – або є курсовий проект

(робота), або ні, але якщо є, то тільки один. Такий саме тип зв'язку буде між типами сутностей Пункт_плану та Навчальна_практика.

Тип зв'язку між типами сутностей Пункт_плану та Заняття також стає умовним.

Додавши типи сутностей Навчальна_практика та Курсовик отримуємо основну ER-діаграма підсхеми «Навантаження викладача та розклад занять» бази даних ІС «Університет» (рис. 2.3).

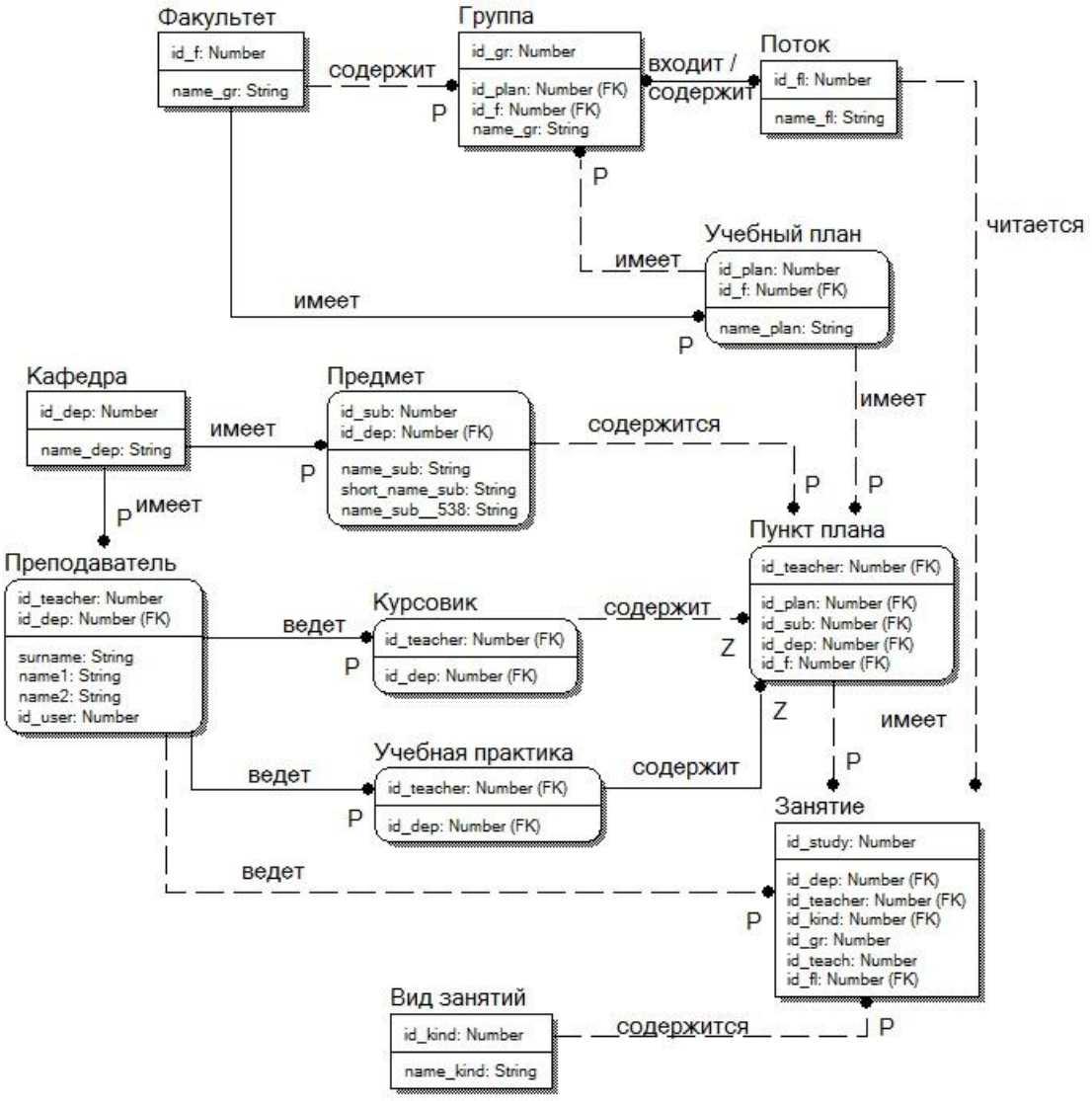


Рисунок 2.3 – Основна ER-діаграма підсхем «Навантаження викладача та розклад занять» бази даних ІС «Університет»

Крім перелічених видів навчального навантаження існують інші види: консультації, перевірка контрольних робіт, керівництво дипломним проектуванням, тощо.

Деякі види навантаження приписані до навчальних дисциплін. Це навантаження розраховується на академічну групу або на кожного окремого студента. Але є види навчального навантаження, які не відносяться до пунктів навчального плану, наприклад, участь викладача у роботі державної екзаменаційної комісії.

Необхідно мати опис видів неаудиторного навантаження – довідник неаудиторного навантаження, новий базовий тип сутності. Також потрібен асоціативний тип сутності для зв'язку деякого виду неаудиторного навантаження з викладачем, що буде виконувати це навантаження.

У кожного пункту навчального плану може існувати текілька видів не аудиторного навантаження, тобто між типами сутностей Пункт_плану та Неаудиторне_навантаження існує тип зв'язку «один до багатьох».

Деякі види неаудиторного навантаження з однієї дисципліни можуть розподілятись між декількома викладачами, кожен викладач може мати декілька видів неаудиторного навантаження з однієї дисципліни – отже у загальному випадку між типами сутностей Неаудиторне_навантаження та Викладач існує тип зв'язку «багато до багатьох».

Викладач може вести декілька видів додаткового навантаження, або не вести жодного.

Деякі види навантаження можуть вести не всі викладачі, наприклад, керівництво науковою роботою магістрів можуть вести тільки доценти та професори, отже у типа сутності Викладач повинен бути атрибут Посада.

Якщо у атрибут Посада (назва посади) додається у список атрибутів типу сутності Викладач, то до списку атрибутів типу сутності Викладач потрібно буде додати ще деякі атрибути, що визначають характеристику викладача в залежності від того, на якій посаді він працює, наприклад, чи може він читати лекції, чи може він виконувати керівництво дипломним проектуванням або магістерською роботу. З точки зору зменшення надмірності даних в базі даних, що проектується, бажано додати у схему базовий тип сутності Посада_викладача. На етапі концептуального проектування введення типу сутностей Посада_викладача є бажаним, але не обов'язковим. Але при логічному проектуванні для реляційної моделі даних введення цього типу сутностей стає обов'язковим з тоски зору нормалізації бази даних.

Виявив типи зв'язків між виявленими типами сутностей, отримуємо доповнену ER-діаграму бази даних – підсхему «Навантаження викладача» (рис. 2.4).

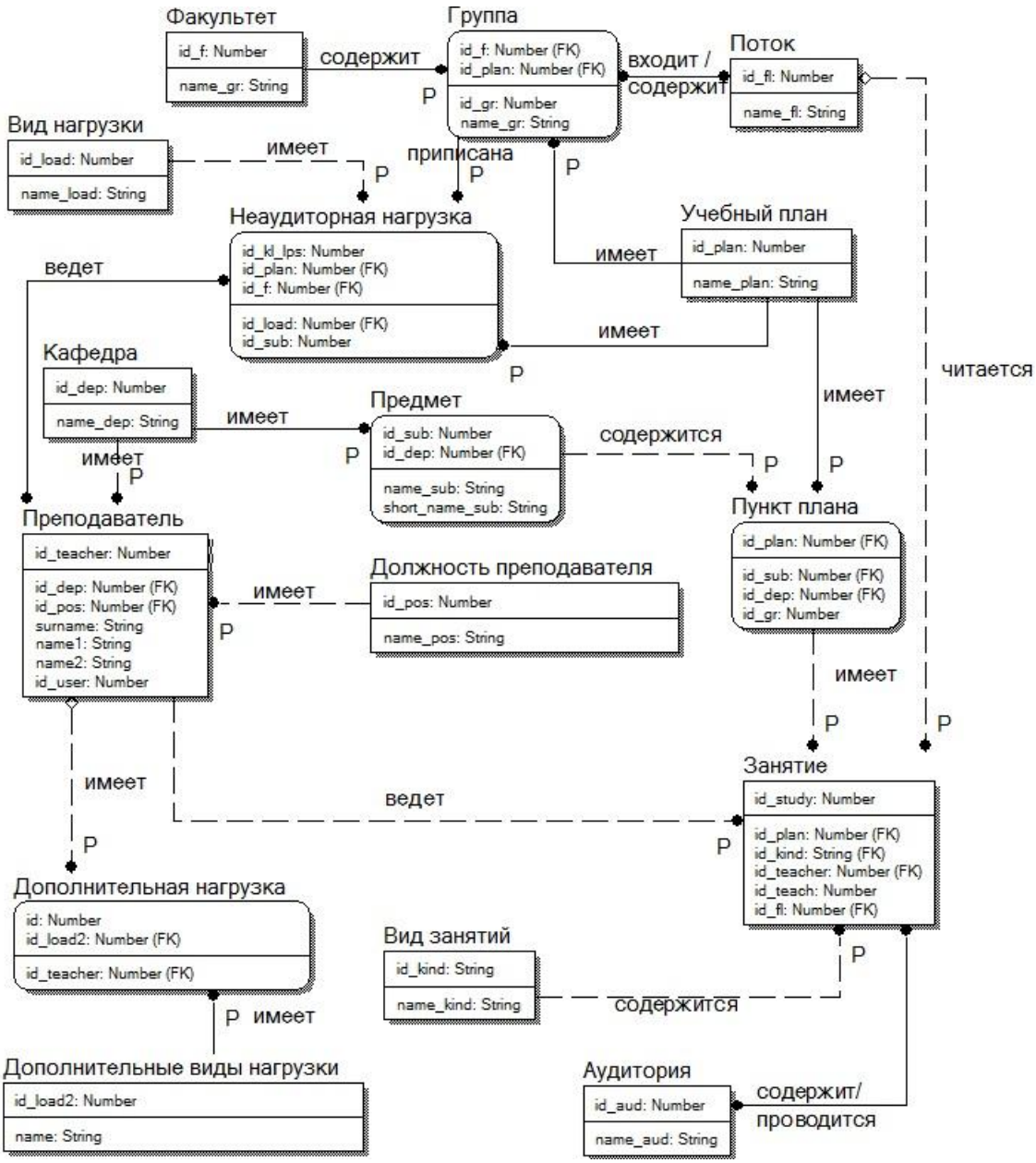


Рисунок 2.4 – Доповнена ER-діаграма підсхеми «Навантаження викладача» бази даних ІС «Університет»

Для отримання об’єднаної концептуальної моделі підсхеми «Навантаження викладача та розклад занять» недоцільно з’єднувати розроблену базу даних з базою даних «Розклад занять».

База даних «Розклад занять» розроблена для складання розкладу занять у одному семестрі одного навчального року, її таблиці мають відповідну структуру. Тобто у типу сутностей Заняття відсутні атрибути Навчальний_рік та семестр. Для бази даних «Розклад занять» розроблена досить велика бібліотека збережених процедур та представлень. Переробки їх недоцільна. Простіше до складу ІС «Університет» включити обидві бази даних. Пе-

ред початком розрахунку розкладу занять потрібні дані копіюються у базу даних «Розклад занять». Після отримання готового розкладу занять на наступний семестр він копіюється у базу даних «Університет».

2.2.2 Визначення атрибутів і зв'язування їх з типами сутностей і зв'язків

Наступним кроком концептуального моделювання є визначення атрибутів і зв’язування їх з типами сутностей і зв’язків.

У типу сутності Факультет будуть атрибути: назва, код факультету (ідентифікатор).

У типу сутності Група будуть атрибути: назва, код групи (ідентифікатор), код факультету, кількість студентів.

У типу сутності Навчальний_план будуть атрибути: назва, код навчального плану (ідентифікатор), код факультету, навчальний рік.

У типу сутності Кафедра будуть атрибути: назва, код кафедри (ідентифікатор).

У типу сутності Посада_викладача будуть атрибути: назва, код посади (ідентифікатор), чи може читати лекції, чи може керувати дипломним проектуванням, чи може керувати науковою роботою магістрів.

У типу сутності Предмет будуть атрибути: код предмета (ідентифікатор), код кафедри, назва, коротка назва.

У типу сутності Викладач будуть атрибути: код викладача (ідентифікатор), код кафедри, прізвище, ім’я, по батькові, код посади, розмір ставки.

У типу сутності Пункт_плану будуть атрибути: код пункту навчального плану (ідентифікатор), код навчального плану, код предмета, кількість годин лекцій, кількість годин семінарів, кількість годин лабораторних занять, кількість годин самостійної роботи, вид семестрового контролю (іспит, залік), наявність курсового проекту (роботи), вид курсового проектування (проект, робота), кількість годин навчальної практики.

У типу сутності Вид_занять будуть атрибути: назва, код виду занять (ідентифікатор).

У типу сутності Вид_навантаження (не аудиторного) будуть атрибути: назва, код виду навантаження (ідентифікатор), чи розраховується на групу (або на студента), розмір навантаження на одну групу (студента).

У типу сутності Заняття будуть атрибути: код заняття (ідентифікатор), код предмету, код виду занять, код групи, код викладача, код потоку (для потокових лекцій), номер підгрупи (для занять з розподілом на підгрупи).

У типу сутності Аудиторія будуть атрибути: назва, код аудиторії (ідентифікатор), номер навчального корпусу.

Більшість виявлених атрибутів відносяться або до домену імен (назв): прізвище, ім’я, по батькові, назва предмета, назва факультету, назва кафедри, назва дисципліни, – або до доменна цілих чисел: номер (код) предмета, код кафедри. Атрибути «чи може …» відносяться до логічного типу. Атрибути «розмір ставки» та «розмір навантаження на одну групу (студента)» відноситься до не цілочисельного типу даних. Вказані типи даних допустимі для СУБД, заснованих на будь-якої з сучасних моделей даних.

2.2.3 Визначення потенційних і первинних ключів

У всіх базових типів сутностей первинним ключем буде атрибут з позначкою (ідентифікатор). У типів сутностей, що за змістом є таблицями-довідниками: Факультет, Кафедра, Предмет, Посада_викладача, Вид_занять, Вид_навантаження, – потенційним ключем (унікальним) є атрибут «назва».

Похідні та асоціативні типи сутностей повинні мати складові потенційні (унікальні) ключі, до складу яких входять атрибути, що є зовнішніми ключами.

2.2.4 Визначення відповідності концептуальної моделі транзакціях користувачів

Для групи користувачів «Працівник кафедри» можна виділити наступні основні транзакції:

1) Отримати дані за навчальними планами, контингентом студентів і прикріпленням груп до навчальних планів.
2) Отримати список дисциплін кафедри на наступний навчальний рік.
3) Визначити списки дисциплін, для яких лекції читаються на потоках і назви потоків.
4) Визначити списки дисциплін і груп, для яких лабораторні заняття або навчальні практики проводяться з розбивкою на підгрупи.
5) Прикріпити викладача до кожного заняття з дисциплін кафедри.
6) Визначити для кожної дисципліни кафедри список видів контролюючих заходів СРС.
7) Визначити академічні групи, в яких керівництво дипломними проектами або магістерськими роботами будуть вести викладачі кафед-

ри, та кількість студентів з цих груп (якщо кафедра керує не всіма студентами групи).

8) Відновити дані про розподіл неаудиторного навчального навантаження викладачів.
9) Оновити дані про допустимі аудиторії для занять.
10) Визначити вимого та побажання викладачів до розкладу занять на наступний семестр.
11) Отримати розклад занять для викладачів кафедри на поточний та / або на наступний семестр.

Група користувачів «Викладач»:

1) Отримати навчальне навантаження на наступний навчальний рік
2) Отримати розклад занять на наступний семестр.

Для всіх зазначених транзакцій у концептуальної моделі визначені сутності, які мають необхідні атрибути.

Таким чином, отримана коректна концептуальна модель бази даних програми.

3 ВИБІР СУБД І СЕРЕДОВИЩА РОЗРОБКИ КЛІЄНТСЬКИХ ДОДАТКІВ

3.1 Вибір СУБД

Для прикладних систем баз даних, що інтегруються у єдину інформаційну систему, була вибрана реляційна СКБД MS SQL Server. В цих прикладних системах баз даних використовувалить програмні застосування лише у вигляді офісних додатків. В інтегрованій ІС «Університет» повинні бути як офісні додатки, так і веб-додатки – сайти кафедр та факультетів. Для розробки інформаційних систем, що орієнтовані не використання лише веб-додатків найчастіше вибирають СКБД MySQL.

У даному випадку бажано залишити СКБД MS SQL Server замість MySQL з тієї причини, що у будуть використовуватись і веб-додатки, і офісні додатки. Тому конкретно вибирати MySQL як основу для сайтів немає сенсу, тим більше, що для розробки сайтів можуть використовуватись такі середовища розробки додатків до баз даних як C ++ Builder та Visual Studio інші, які вже були використані для розробки додатків до прикладних баз даних.

В розроблених системах бах даних накопичена велика бібліотека серверного програмного забезпечення – представлень і збережених процедур, які абсолютно нераціонально буде заново переписувати під MySQL, враховуючи, що MySQL під офісні додатки практично не використовується. Розробляти додатки до ІС «Університет» можна в будь-якому середовищі (C ++ Builder, Delphi, Visual Studio) і на будь-якій мові (C ++, C #, Visual Basic). Можлива розробка сайтів на будь-якій мові.

Система SQL Server 2008 дозволяє звертатися до даних з усіх програм, розробленого із застосуванням технологій Microsoft .NET і Visual Studio, а також у межах сервісно-орієнтованої архітектури та бізнес-процесів – через Microsoft BizTalk Server. Співробітники, відповідальні за збір та аналіз інформації, можуть працювати з даними, не покидаючи звичних додатків, якими вони користуються щодня, наприклад додатків випуску 2007 системи Microsoft Office. SQL Server 2008 дозволяє створити надійну, продуктивну, інтелектуальну платформу, що відповідає всім вимогам по роботі з даними.

Одним з основних недоліків SQL Server 2005 була відсутність механізму автоматичного заповнення та можливості налагодження кодів T-SQL. Підтримка технології IntelliSense в версії SQL Server 2008 (вона реалізована в редакторі Query Editor середовища SQL Server Management Studio, SSMS) до-

зволяє здійснювати перевірку синтаксису кодів T-SQL і реалізувати автоматичне призначення імен об'єктів бази даних. Помилки в синтаксисі мови T-SQL позначаються червоною хвилястою відміткою в кінці кожного оператора коду T-SQL.

При роботі зі службою SQL Server 2005 для налагодження сценаріїв T-SQL раніше доводилося використовувати пакет Microsoft Visual Studio. Редактор Query Editor, що входить до складу SQL Server 2008, дозволяє проводити повноцінну налагодження коду T-SQL і дає можливість встановлювати точки переривання, тому використовувати Visual Studio при роботі зі сценаріями T-SQL більше не була необхідною. Іншим важливим поліпшенням редактора Query Editor є можливість згортати і розгортати блоки коду в вікні редактора. Однак нові функції можуть використовуватися тільки при роботі з базами даних SQL Server 2008. Якщо підключити додаток Query Editor до бази даних SQL Server 2005, ви не зможете активувати механізми IntelliSense, будь то налагодження або структурування коду. До того ж ці механізми не входять до складу набору SSMS Basic, використовуваного для управління версіями SQL Server 2008 Express і SQL Server 2008 Express with Advanced Services.

Новий механізм стиснення даних цілком "прозорий" з точки зору клієнтських додатків і для його роботи не потрібно вносити зміни до додатків. Стиснення даних в SQL Server 2008 може знизити розмір збережених на диску баз даних і прискорити процеси виконання резервного копіювання та відновлення шляхом зменшення кількості операцій введення / виводу.

Механізм TDE розширює можливості шифрування служби SQL Server 2005. Механізми SQL Server 2005 здійснюють шифрування даних на рівні осередків. Однак управління ключами шифрування здійснюється вручну, а для доступу до зашифрованих даних необхідно міняти код додатків. Механізм TDE дозволяє здійснювати шифрування бази даних цілком, при цьому процес шифрування повністю прозорий для додатків кінцевих користувачів.

Підтримка технології Hot Add CPU дозволяє SQL Server 2008 розпізнавати і використовувати процесори, додані в систему, без необхідності перезавантажувати сервер або його служби. Дана технологія доповнює можливість «гарячого» додавання оперативної пам'яті, Hot Add RAM, роблячи платформу SQL Server 2008 ідеальним рішенням для установки в віртуальних середовищах і для участі в динамічному управлінні робочим навантаженням.

Нововведенням пакета SQL Server 2008 є інструмент Resource Governor, який дозволяє керувати кількістю системних ресурсів, що виділя-

ються платформою для заданої робочої навантаження. Наприклад, Resource Governor дозволить обмежити число системних ресурсів, що виділяються на обробку некоректно складених запитів користувачів, які в іншому випадку могли б помітно знизити продуктивність системи. Крім того, даний інструмент зробить більш передбачуваними процеси виконання запитів і завдань в системі.

При роботі з типом даних DATETIME, використовуваним в ранніх версіях SQL Server, доводилося об'єднувати дані про час і дату в одному стовпці. Нові типи даних DATE і TIME, запропоновані в SQL Server 2008, дозволяють зберігати значення дати і часу окремо, а для зберігання більш точних значень цих параметрів створені додаткові типи. Дані класу FILESTREAM забезпечують високошвидкісний доступ до великих об'єктів (LOB). Платформа SQL Server 2008 також пропонує використовувати нові просторові типи даних, на основі яких можна створювати додатки зіставлення.

SQL Server 2008 містить поліпшення мови T-SQL. Оператор DECLARE тепер дозволяє визначати значення змінних. Нові структурні оператори присвоювання (+ =, - =, / =,% =, & =, | =, and ^ =) дозволяють більш ефективно застосовувати логічні оператори XOR, XAND, а також оператори конкатенації рядків. Нова функція конструктора стовпців дає можливість задіяти оператор INSERT для вставки декількох стовпців. Новий оператор MERGE дозволяє виконати функції INSERT, UPDATE і DELETE за один крок. Ще одна нова особливість мови T-SQL – параметри з табличними значеннями – дозволяє передавати таблицю як параметр в збережену процедуру або функцію. Табличні параметри корисні, коли необхідно передати в спричинюється процедуру (функцію) велика кількість даних. Крім того, нова функція GROUPING SETS забезпечує додатковий контроль результатів і механізмів обробки, використовуваних в запитах SQL Server.

Одне з найбільш важливих змін, які торкнулися системи SSRS, полягає в тому, що тепер не потрібно попередньо встановлювати інструментарій Microsoft IIS. Крім того, був істотно змінений зовнішній вигляд конструктора Report Designer. Також зверніть увагу на новий формат звітів Tablix, що поєднує атрибути таблиці і звіти у вигляді матриць. Ще одна особливість, яка допоможе кінцевим користувачам отримувати доступ до даних в системі SQL Server 2008, – можливість створювати звіти в додатках Microsoft Excel і Word. Компанія Microsoft придбала ліцензію на використання пакета OfficeWriter від компанії SoftArtisans. Цей додаток здійснює інтеграцію редакторів Excel і Word зі службами SQL Server і базами даних SSAS.

Механізм управління на основі політик дозволяє адміністратору бази даних застосувати корпоративні стандарти, такі як налаштування бази даних і угоди про іменування об'єктів, до декількох серверів одночасно.

Для адміністраторів баз даних додаткові функції управління роблять сервер SQL Server 2008 новим відмінним продуктом. Нове управління політиками, можливість декількох запитів серверу, сервери конфігурації і збирач даних/сховище управління надають нові потужні можливості для адміністраторів баз даних, які часто відповідальні за управління великими і складними середовищами баз даних з сотнями або тисячами баз даних на десятки або навіть сотні серверів.

Функція управління політиками SQL Server 2008, яка називалася функцією динамічного управління (DMF – Declarative Management Framework) в CTP-версіях, дозволяє створювати і виконувати політики конфігурації для одного або декількох серверів баз даних. Ці політики забезпечують застосування параметрів стандартної конфігурації на всіх цільових серверах і базах даних. Приклад цієї функції показаний на рис. 3.1.

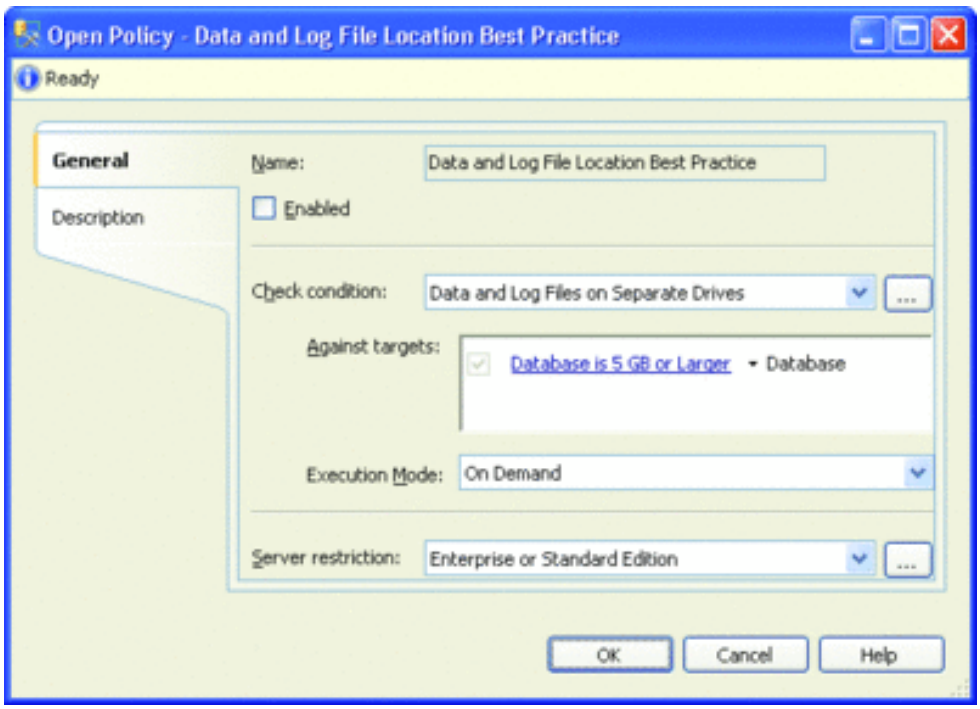


Рисунок 3.1 – Політика рекомендацій з розміщення файлів даних і журналів

Політики створюються на основі визначеного набору аспектів. Кожен аспект містить підгрупу параметрів конфігурації SQL Server 2008 та інших подій, якими можна управляти. Даним аспектам зіставляються умови для

створення політики. Умови – це значення, дозволені для властивостей аспекту, параметрів конфігурації або інших подій, що містяться в даному аспекті.

Умовами також є значення, використовувані для фільтрів політики. Припустимо, політику необхідно застосувати тільки для певної бази даних. У цьому випадку можна створити умову, що містить ім'я бази даних, а потім додати цю умову до політики. Після цього політика буде застосовуватися тільки до цій базі даних.

Нові можливості взаємодії декількох серверів і сервери конфігурації зручні при необхідності одночасного виконання запитів до декількох серверів. Можна зареєструвати сервери в середовищі Management Studio, а потім об'єднати їх за допомогою групування. При необхідності виконання політики або запиту для всіх серверів в групуванні її слід просто натиснути правою кнопкою миші і виконати необхідні дії.

Додатковою перевагою є можливість налаштування цієї функції для повернення одного набору результатів для сервера або об'єднання наборів результатів в один великий набір. Також можна вказати, чи будуть імена сервера і баз даних бути частиною результатів, щоб відокремити результати для окремих серверів. Можливість збереження зареєстрованих серверів на сервері конфігурації, а не в окремих Management Studio є великою перевагою.

3.2 Вибір середовища розробки клієнтського додатки до БД

Продукти Microsoft для розробників давно входять до списку найбільш затребуваного програмного забезпечення для програмістів різного рівня. Visual Studio 2010 створювався з включенням елементів спільної роботи і обміну даними між програмістами, зайнятими в проекті. Для цього потрібна організація централізованого сховища інформації з гнучким механізмом розмежування доступу до контенту, наочними інструментами контролю стану проекту та участі програмістів в досягненні ключових показників, системою відстеження змін, які внесені в код і чекають схвалення.

Visual Studio 2010 є блискучою інтегрованим середовищем розробки (IDE) на базі WPF. Тепер в ній з'являються різні засоби, які раніше були доступні лише як додаткові продукти, сюди відносяться R # і Refactor. Так само з приводу середовища розробки можна помітити появу сніпетів (іноді званих фрагментами або шаблонами), створення заготовок класів, ієрархії викликів і швидкий пошук.

Visual Studio 2010 являється гнучким і розширюваним продуктом. Велика частина середовища розробки тепер створена із застосуванням WPF і може бути налаштована засобами MEF (каркас керованого розширення).

Сама мова також змінилася, що дає можливість писати більш ясний програмний код, серед даних змін можна відзначити необов'язкові і іменовані параметри, зміни в варіантності і динамічну функціональність.

У платформі .NET (дот нет) є технології, які не змінилися радикально в даному випуску, але були прибрані багато помилок і була проведена деяка доробка. До таких технологій відноситься технологія asp.net.

Так само з приводу програмування для веб можна помітити, що були додані бібліотеки jQuery і Microsoft Ajax.

У цьому випуску каркаси Windows Workflow Foundation (WF) і Windows Communication Foundation (WCF) набагато тісніше інтегровані один з одним. Каркас WF був радикально поліпшений за рахунок помітного поліпшення візуального конструктора, який підтримує нові дії і полегшену настройку. А каркас WCF тепер став простіше у використанні, також тепер включає в себе нову функціональність служби виявлення.

У WPF теж були внесені зміни, до яких можна віднести підтримку поліпшеного візуально конструктора, технології мультитач, які використовуються для сенсорних екранів і панель завдань Windows 7.

Стався зсув у бік багатоядерних процесорів – на даний момент створюються багатоядерні процесори і соответсвенно повинні створюватися програмні продукти з урахуванням цього важливого фактора. Написання паралельно працюючих програм є не легкою справою, але завдяки даному випуску ця задача помітно стала легше, тому стали доступні можливості розпаралелювання та засоби налагодження.

3.2.1 Основні можливості та особливості Microsoft Visual Studio 2010

У сімействі продуктів Visual Studio використовується єдина інтегрована середовище розробки (IDE), що складається з декількох елементів: рядка меню, панелі інструментів Стандартна, різних закріплених або автоматично приховуються вікон інструментів в лівій, нижньої або правої областях, а також області редакторів. Доступні вікон інструментів, меню та панелей інструментів залежить від типу проекту або файлу, в якому виконується розробка.

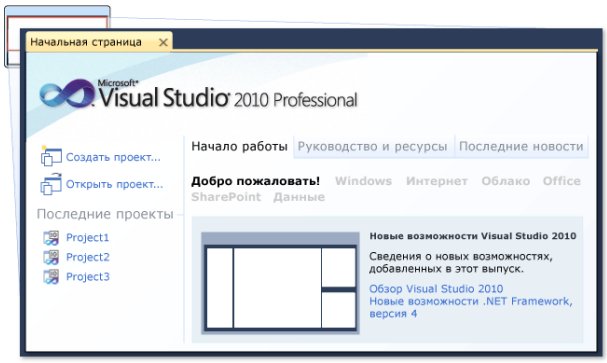


Рисунок 3.2 – Інтегроване середовище розробки, в якій встановлені загальні параметри розробки

Частина проектів могла бути створена в старших в Microsoft Visual Studio і з використанням попередніх версій мов .NET. В цьому випадку необхідно згадати про можливість використання цих даних і в Microsoft Visual Studio 2010 – вони будуть автоматично сконвертовані відповідно до оновлених компонентами (зокрема, буде замінена версія .NET) і вже після цього відкриті в IDE і інтегровані з новими компонентами. Проте, можлива й зворотна операція, коли нову систему необхідно інтегрувати зі старим кодом – це все можна виконати у відповідному діалоговому вікні налаштування. У середовищі розробки є повноцінна підтримка можливостей інтерфейсу Windows 7 (мультитач-управління, графічні ефекти оболонки Aero, «стрічковий" інтерфейс і так далі) і багатопроцесорних систем (підтримка створення багатопоточних додатків).

Помітно спростилися і покращилися в плані призначеного для користувача інтерфейсу інструменти візуалізації коду – дизайнери і програмісти бачать звичний кожним з них інтерфейс (виконаний з використанням Windows Presentation Foundation і Silverlight; для підвищення зручності сприйняття, зокрема, з нього прибрані деякі лінії і градієнти прибрані, щоб знизити захараженість панелей), при цьому завжди є можливість перемикання між режимами. З істотних нововведень варто відзначити підтримку мультимоніторних систем – це зручно для налагодження коду. Крім того, з вікна середовища розробки можна завантажувати, встановлювати, спільно використовувати і керувати шаблонами, пакетами і компонентами. Хоча і візуально, і функціонально новий інтерфейс максимально нагадує колишній, він сильно відрізняється технологічно – тепер для промальовування робочої області замість Windows Forms використовується WPF. Це не тільки додало зовнішнього лиску, але і дозволило реалізувати безліч удосконалень, спрямованих

на підвищення продуктивності роботи, в тому числі і з урахуванням зрослих можливостей продукту. Наприклад, у вікні вибору типу проекту, а також в інспектора властивостей з'явилися поля, відфільтровує невідповідні елементи зі зворотним відліком потрібного назви; редактор тепер дозволяє масштабувати текст колесом прокрутки миші, утримуючи клавішу Ctrl так само, як це робиться в браузері; покращилася підтримка декількох моніторів; при розробці із застосуванням модульних тестів буде корисна нова функція, що відображає в реальному часі тести, на результат яких можуть вплинути поточні зміни вихідного коду. Був значно перероблений і візуальний редактор WPF / Silverlight – тепер його можливості впритул наблизилися до Expression Blend, що навіть дає підстави виключити останній з набору інструментів, якщо для візуальної розробки на базі XAML застосовується Visual Studio 2010 року.

Відладчик тепер доповнений функцією, що отримала назву «історичне налагодження» (Historical Debugging). Традиційно для точного виявлення джерела помилки застосовуються точки зупину і покрокове виконання коду до тих пір, поки не виникне виняток. Функція історичного налагодження дозволить радикально спростити даний процес – при її активації всі дії програми, аж до виклику методів і обробки подій, відстежуються і зберігаються. Таким чином, у багатьох випадках для пошуку помилки взагалі немає необхідності використовувати точки зупину, досить перервати роботу програми за допомогою відладчика і в спеціальному вікні переглянути історію виконання (за винятком значень локальних змінних, збереження яких не передбачено з міркувань продуктивності). Потім стан програми можна покроково перегортати вперед-назад («минуле» позначається спеціальним значком – стрілкою годинника) або відразу переходити до будь-якої потрібної точки. Оскільки активація цієї функції для збереження всієї історії роботи програми може викликати значне додаткове навантаження на систему, передбачені можливість обмеження максимального обсягу протоколів і гнучке налаштування відслідковуються методів і подій.

Звичайно, що існуючі пертурбації вимагають тестування на предмет забезпечення повного переходу між версіями. Що з'явилося в продукті вікно Test Impact View відображає всі впливу змін в коді на тестування проекту – розробник зможе побачити, які тести йому потрібно виконати після того чи іншого впровадження або виключення фрагмента, перемикаючись швидко між самим кодом і списком тестів. В результаті стає досить просто відстежити, які саме розробники допускають помилки, в чому вони полягають і наскільки вони критичні. У Team Foundation Server 2010 з'явилася відповідна за-

сіб, яке дозволяє керівнику проекту реєструвати надходять від програміста зміни в коді – відповідно, якщо вони прийняті або не прийняті, він отримує про це повідомлення. Таким чином зменшується до мінімуму ризик збоїв при складанні, якщо проблема виникла на якомусь з етапів. Крім усього іншого, є і спеціальні інструменти налагодження і профілювання створених багатопоточних додатків.

Не менше значення в Microsoft Visual Studio 2010 приділено і актуальною тенденції ринку ПО, як віртуалізації. У середовищі можна створити кілька віртуальних середовищ з декількома віртуальними ПК, на яких проводити тести, аналізи, збірки і розгортання додатків. Система управління лабораторією тестування побудована на базі «рідний» для Microsoft System Center Virtual Machine Manager, що зводить до мінімуму витрачається час на віртуалізацію розробки. Крім того, Microsoft Visual Studio 2010 сумісна з фірмовою «хмарної» платформою Azure.

У Visual Studio 2010 реалізовані два концептуальних підходи до ведення проекту – лінійний і за допомогою гнучких спринтів (декількох етапів, що включають в себе список встановлених заходів щодо виконання проекту). Для цього Visual Studio 2010 включає новий набір типів робочих елементів, типів зв'язків, панелі моніторингу, звіти і документи, які більше відповідають стилю роботи груп, які використовують гнучкий процес. У керівників проекту з'являються всі необхідні інструменти, що дозволяють визначати і перемикати навантаження на конкретного програміста / групу програмістів, а також, що важливіше, вибудовувати ієрархію відносин і залежностей між завданнями. Відповідно, з'являється можливість швидкого перемикання між списками завдань з виявленням завантажених ділянок роботи і швидкого перенесення запланованих робіт на інший час без шкоди для загального термінів виконання проекту. Заздалегідь встановлена схема ієрархії може бути багаторазово повторена в ітераціях самого проекту і тому помітно (до двох разів) заощадить час, що витрачається на повторне розподілення ролей в проекті після внесення змін до нього.

Нова можливість IntelliTrace дозволяє залишити проблему неможливості відтворення помилок у минулому. Тестери можуть детально і ефективно описувати помилки і пов'язані системні відомості і навіть включати до звіту моментальні знімки середовища. Таким чином розробники завжди зможуть відтворити їх у тому стані, в якому їх виявили.

Architecture Explorer допоможе розпізнати активи існуючого коду та їх взаємозалежності. Можна створити точні моделі структури програми та навіть заглибитися в певні області для більш повного аналізу.

Для визначення і зв'язування логічної архітектури додатку та перевірки параметрів коду на відповідність необхідної архітектурі можна використовувати структурні діаграми. Структурні діаграми дозволяють стежити за відповідністю вимогам під час розробки.

Visual Studio Test Professional 2010, частина середовища Visual Studio 2010 Ultimate, надає єдиний засіб для запису та оновлення тестових вимог, автоматизації ручних тестів і прискорення циклу виправлення і схвалення додатки за рахунок охоплення всього контексту тестування. Це дає розробникам все, що необхідно для відтворення будь-якої помилки.

Visual Studio 2010 дозволяє здійснювати управління життєвим циклом додатків (ALM). Створення успішних програм увазі чіткий і безперебійний процес зручний для всіх учасників робочої групи. Вбудовані в Visual Studio 2010 Ultimate кошти ALM допомагають компаніям організувати ефективну спільну роботу і систему комунікації на будь-якому рівні, зробити видимим фактичний стан проекту, забезпечуючи доставку високоякісних рішень з меншими витратами.

Visual Studio 2010 Ultimate оснащена всіма удосконаленими засобами тестування для забезпечення стабільної якості коду. Розробник може використовувати кодовані тести інтерфейсу користувача, за допомогою яких можна автоматизувати тестування інтерфейсу веб-додатків і додатків Windows, засоби ручного тестування, функцію Test Professional, тестування продуктивності веб-додатків, тестування навантаження, і іншими корисними функціями, які недоступні в інших версіях Visual Studio.

Architecture Explorer в Visual Studio 2010 Ultimate допоможе розпізнати активи існуючого коду та їх взаємозалежності. Структурні діаграми забезпечують архітектурну узгодженість і дозволяють оцінювати код. Крім того, Visual Studio 2010 Ultimate підтримує п'ять основних видів UML-діаграм, які використовуються разом з кодом.

Visual Studio 2010 Ultimate з MSDN є найбільш повним пропозицією для розробників. Крім можливостей, що надаються Visual Studio 2010 Professional з MSDN і Visual Studio 2010 Premium з MSDN, Ultimate з MSDN включає в себе додаткове процесорний час Windows Azure, доступ до Team Foundation Server за допомогою Microsoft Visual Studio Team Explorer

Everywhere 2010 без використання Visual Studio та програмне забезпечення Test and Lab Manager.

Visual Studio 2010 дозволяє реалізувати всі ідеї на різних платформах, включаючи Windows, Windows Server, веб-додатки, хмарні системи, Office, SharePoint та інші, за допомогою одного середовища розробки.

Team Foundation Server (TFS) – це платформа спільної роботи на основі рішення Майкрософт з управління життєвим циклом додатків. TFS автоматизує і спрощує розробку програмного забезпечення, а також забезпечує можливість відстежувати всі елементи проекту і бачити в реальному часі його стан для всіх учасників проекту. Платформа також надає потужні засоби панелі моніторингу та створення звітів.

Розробка баз даних вимагає тієї ж ретельності і уваги що й розробка додатків. У Visual Studio 2010 Ultimate це враховується: користувачам надаються надійні засоби розробки та управління змінами, які дозволяють синхронізувати додаток і базу даних.

4 ЛОГІЧНЕ І ФІЗИЧНЕ ПРОЕКТУВАННЯ БД

У другому розділі було виконано концептуальне проектування бази даних проекту, що розроблюється. У третьому розділі в якості моделі даних була вибрана реляційна модель. Тому логічне проектування необхідно виконати для цієї моделі.

4.1 Логічне проектування

Розроблена в другому розділі модель вже майже є логічною моделлю реляційної бази даних. Як видно з ER-моделі, представленої на рис.2.4, ніде немає розривів зв'язків, але має декілька зв'язків типу «багато до багатьох», які не допустимі у реляційної бази даних.

Для відповідності логічної моделі БД вибраній реляційній моделі даних необхідно ввести три асоціативні сутності, які дозволять перетворити три типу зв'язку «багато до багатьох» у шість типів зв'язку «один до багатьох».

Тому вводяться такі асоціативні сутності:

FlowGroup (Склад потоку) для зв'язків між типами сутностей Група та Потік. У цього типу сутності будуть атрибути: ig_gr, id_fl.

StudyAuditory (Аудиторії для занять) для зв'язків між типами сутностей Аудиторія та Заняття. З атрибутами: id_study, id_aud, prioritet.

TeachKindLoad (Вид навантаження викладача) для зв'язків між типами сутностей Викладач та Неаудиторне_навантаження з атрибутами: id_teacher, id_kl_lps, count_unit.

Таким чином, набір сутностей, визначених при концептуальному проектуванні, перетвориться в набір відношень при логічному проектуванні по реляційній моделі. Отримуємо ER-діаграму логічної моделі «Навантаження викладача та розклад занять» БД «Університет», яка показана на рис. 4.1 та в додатку А.

Після усунення з моделі зв'язків типу «багато до багатьох» відношення задовольняють правилам нормалізації.

Незалежно від того, що обрана СУБД SQL Server 2008 дозволяє використовувати ідентифікатори з кирилицею, зручніше всеж-таки писати програмне забезпечення не перемикаючись на інші мови. Тому бажано використовувати назви відношень, так само і назви атрибутів відношень з латиницею. Імена відношень бази даних вказані в таблиці 4.1

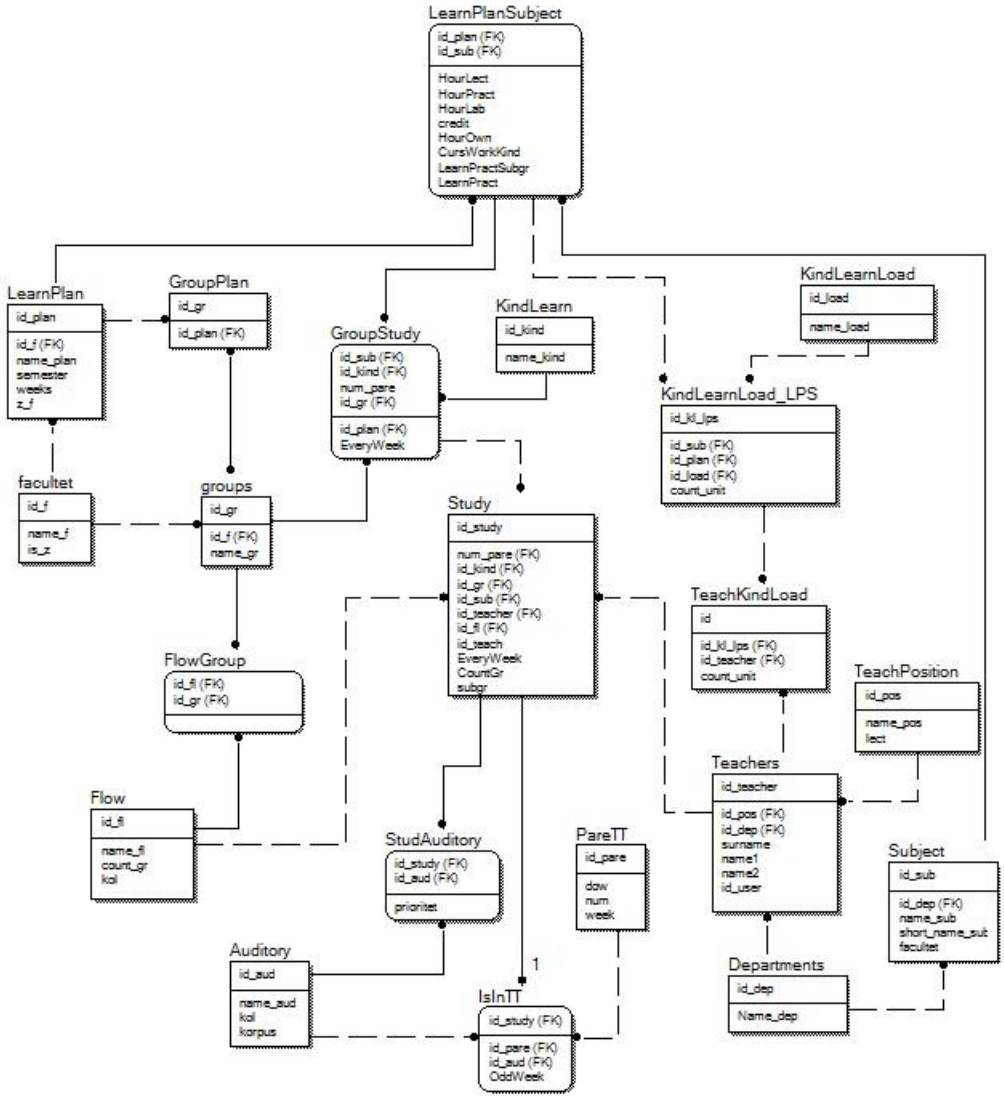


Рисунок 4.1 – ER-диаграмма логічної моделі бази даних «Університет»

Таблиця 4.1 – Імена відношень бази даних «Університет»

Відношення	Назва англійською
Факультет	Facultet
Група	Groups
Потік	Flow
Навчальний план	LearnPlan
Вид навантаження	KindLearnLoad
Неаудиторне навантаження	KindLearnLoad_LPS
Кафедра	Departments
Предмет	Subject
Пункт плану	LearnPlanSubject
Викладач	Teachers

Продовження табл. 4.1.

Відношення	Назва англійською
Посада викладача	TeachPosition
Заняття	Study
Вид заняття	KindLearn
Аудиторія	Auditory

Перевірку відповідності відношень вимогам призначених для користувача транзакцій робити не має сенсу, оскільки логічна модель практично співпадає з концептуальною моделлю, для якої вказана перевірка була виконана.

Вимоги підтримки цілісності даних включають стандартні вимоги цілісності сутностей і посилальної цілісності, які реалізуються у фізичній моделі при створенні первинних і зовнішніх ключів таблиць.

Таким чином, отримана логічна модель бази даних. Логічна схема БД приведена в додатку А.

4.2 Фізичне проектування

При виконанні фізичного проектування, як вказувалося в другому розділі, після вибору конкретної СУБД необхідно спроектувати базові відношення в середовищі цієї СУБД; спроектувати похідні відношення; реалізувати обмеження цілісності; визначити індекси; розробити представлення для кожної групи користувачів.

При проектуванні таблиць необхідно вирішити питання про доцільність використання в таблицях первинних ключів з властивістю автоматичної нумерації. Подібні первинні ключі логічно використати для таблиць-довідників, вміст яких рідко оновлюється, наприклад, таблиць, що містять список кафедр, або список факультетів, список посад викладачів.

В існуючій інформаційній системі використовуються первинні ключі, що розраховуються, для багатьох таблиць.

Ієрархічний первинний ключ в підпорядкованих таблицях повинен обчислюватися з використанням первинного ключа батьківської таблиці (таблиці більш високого рівня ієрархії), тобто по зовнішньому ключу самої підпорядкованої таблиці за формулою:

$$ID\_LOWER = ID\_UPPER \cdot 2^N + code, \qquad (4.1)$$

де ID_UPPER – значення зовнішнього ключа (первинного ключа таблиці попереднього рівня ієрархії);

code – порядковий номер запису в групі записів з однаковим значенням зовнішнього ключа, наприклад, номер групи усередині факультету;

N – кількість біт в записі значення code.

Інших індексів окрім первинних унікальних ключів не буде, при цьому в існуючій базі даних вже є обчислювані ключі, у яких в вершині ієрархії стоять факультет і кафедри. Тобто, від факультетів відраховуються id студентів, id, групи, id планів, від кафедри відраховується id викладача, id дисциплін і відповідно далі пов'язані з ними ключі.

Також бажано ввести ще один обчислюваний первинний ключ для таблиці GroupLoad, в яку обчислюється і заноситься навантаження (як аудиторні, так і неавдиторний) групи з різних видів занять (модулі, курсові, іспити, заліки і т.д.).

В таблиці GroupLoad є дуже складний потенційний ключ, який складається з п'яти атрибутів (LearnYear, autumn, id_sub, id_group, id_kind). Якщо цей ключ зробити первинним, то будуть досить складні запити з підлеглих таблиць таблиці GroupLoad. У цій таблиці будуть підлеглі таблиці при розподілі навантаження між викладачами. Навантаження розраховується для дисципліни, але його ще потрібно розподіляти між викладачами. З таким складовим ключем незручно далі робити підлеглі таблиці. Тому вказаний потенційний ключ вибирається у якості унікального ключа, і в таблицю додається обчислюваний первинний ключ, якій розраховується по складових унікального ключа.

Отримуємо наступну бітову маску обчислюваного первинного ключа у таблиці GroupLoad (рис. 4.2):

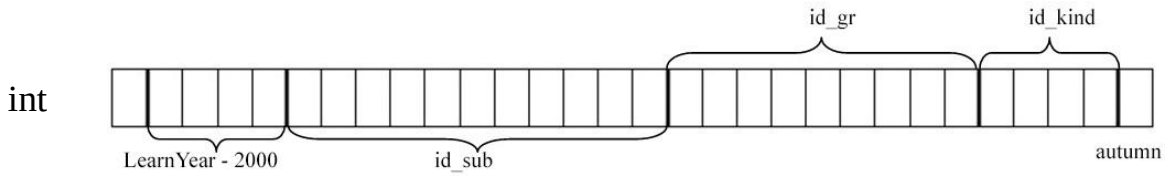


Рисунок 4.2 – Розподіл бітів запису зовнішнього ключа і порядкового номера запису в первинних ключах таблиці GroupLoad

Відповідно до логічної моделі, спроектованої в розділі 4.1 і приведеної в додатку Б, для вибраної СУБД MS SQL Server 2000 можна спроектувати фі-

зичну базу даних. Для цього для кожного атрибуту треба вказати домен, тобто область допустимих значень, і відповідно до домена вибрати найбільш відповідний тип для атрибуту.

У додатку В наведена фізична схема бази даних «Університет» для нових відношень, яких не було у попередніх базах даних «Навантаження викладача» та «Розклад занять».

У фізичну схему бази даних також входять представлення. Для реалізації уявлень користувача на вигляд таблиці навчального навантаження потрібно створити декілька представлень бази даних. Ця необхідність зумовлена невідповідністю структури таблиці в поданні користувача і структури таблиці, використаної в базі даних для зберігання відповідних даних.

В уявленні користувача навантаження з дисципліни повинно записуватись у один рядок таблиці зі стовпцями, в яких вказуються назва дисципліни, назва групи (потоку), розміри навантаження за різними видами: лекції, семінари, консультації, лабораторні заняття, і.т.д. В базі даних є таблиці-довідними для навчальних дисциплін, груп, потоків, видів навантаження. Розмір навантаження записується у один стовпець (а не в рядок), з визначенням за допомогою відповідних ідентифікаторів, до якої дисципліни, групи (потоку) та виду навантаження відносяться ці дані.

Для перетворення стовпця даних у рядок потрібно створити два представлення бази даних. Перше представлення записує стовпець даних у вигляді діагоналі матриці – всі недіагональні елементи матриці дорівнюють нулю. Друге представлення за допомоги вибірки з групуванням перетворює діагональну матрицю даних у рядок даних.

У додатку Г наведені представлення бази даних.

5 СЕРВЕРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

У базі даних «Расписание» має бути серверне ПЗ у вигляді набору збережених процедур, що забезпечують маніпулювання даними в таблицях бази даних: додавання записів, оновлення, видалення, – а також для необхідної обробки даних.

Процедура FillDepSubLoadHourTemp формує таблицю з аудиторним навантаженням кафедри за вказаний навчальний рік:

CREATE proc DoDepSubLoadTemp

@id_dep smallint,

@LearnYear smallint

Параметри процедури:

@id_dep – ідентифікатор кафедри;

@LearnYear – номер навчального року (номер року для осіннього семестру навчального року).

Процедура FillDepSubLoadHourTemp формує для заданого навчального року таблицю наявності по всім дисциплінам кафедри різного виду занять: лекцій, практичних занять (семінарів), лабораторних занять, – для всіх академічних групп, що вивчають дану дисципліну.

Якщо для деяких груп в базі даних зберігається інформація, що з деякої дисципліни лекції читаються на потоці, то для цієї лекції цієї групи з цієї дисципліни у вихідну таблицю записується назва потоку академічних груп.

Лабораторні заняття можуть проводитись з розподілом на підгрупи. Ля кожного лабораторного заняття кожної групи з кожної дисципліни у вихідну таблицю записується, чи розподіляється вона на підгрупи. Дані про розподіл груп на підгрупи для лабораторних занять по дисциплінах зберігаються у таблиці IsSubGr.

Процедура FillDepSubLoadHourTemp формує таблицю з повним навчальним навантаженням кафедри за вказаний навчальний рік:

CREATE proc DoDepSubLoadTemp

@id_dep smallint,

@LearnYear smallint

Параметри процедури:

@id_dep – ідентифікатор кафедри;

@LearnYear – номер навчального року (номер року для осіннього семестру навчального року).

Процедура FillDepSubLoadTemp формує для заданого навчального року таблицю зі значеннями величини навантаження (у академічних годинах) по всім дисциплінам кафедри для всіх видів навчального навантаження для всіх академічних групп, що вивчають дані дисципліну.

В навчальне навантаження з дисципліни включаються аудиторні заняття, консультації, прийом екзаменів та заліків, керівництво курсовими роботами та проектами, виконання контролю самостійної роботи студентів, керівництво навчальною практикою.

Процедура FillGroupLoadDep додає у таблицю GroupLoad записи пронавантаження по всім дисциплінам кафедри від виконання контролю самостійної роботи студентів:

```
CREATE proc FillGroupLoadDep
@id_dep tinyint,
@LearnYear smallint,
@k tinyint output,
@msg varchar (100) output
```

Параметри процедури:

@id_dep – ідентифікатор кафедри;

@LearnYear – номер навчального року (номер року для осіннього семестру навчального року);

@k – вихідний параметр, код помилки;

@ msg – вихідний параметр, повідомлення про помилку.

В процедурі FillGroupLoadDep визначаються всі навчальні плани, які використовуються для навчального року з номером @LearnYear, в яких є дисципліни, закріплені за кафедрою з ідентифікатором @id_dep. Для всіх цих дисциплін, що вивчаються згідно знайдених навчальних планів, визначається, чи задані контролюючі заходи для контролю самостійної роботи студентів. Якщо ці заходи задані, розраховується навчальне навантаження кафедри від перевірки виконання студентами завдань на СРС.

Якщо контролюючі заходи для контролю самостійної роботи студентів задані не для всіх дисциплін кафедри, виводиться відповідне повідомлення про помилку завдання даних.

Процедура ModulWPEdit оновлює запис у таблиці ModulWP:

```
create procedure ModulWPEdit
  @id_modWP int,
  @id_wp int,
  @name_modWP varchar (50),
```

@type bit,
@bal_percent smallint,
@k smallint output,
@msg varchar (100) output

Параметри процедури – ті ж самі, що і в процедурі ModulWPAdd.

Для модуля робочої програми, код якого дорівнюєid_modWP, процедура ModulWPEdit привласнює нові значення атрибутів name_modWP, type, bal_percent. Якщо змінюється в модулі назву і модуль з такою назвою вже є в робочій програмі, генерується повідомлення про помилку.

Процедура ContentModulWPAdd додає в таблицю ContentModulWP запис:

create procedure ContentModulWPAdd
@id_modWP int,
@id_test smallint,
@id_mod smallint,
@per_cent smallint,
@type bit,
@k smallint output,
@msg varchar (100) output

Параметри процедури:

@id_modWP – код модуля робочої програми;

@id_test – код комп'ютерного тесту;

@id_mod – код модульної контрольної;

@per_cent – % від балу за модуль робочої програми;

@type – комп'ютерний тест чи ні;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура ContentModulWPAdd перевіряє чи включений вже в даний модуль робочої програми комп'ютерний тест з кодомid_test (якщо id_test! = Null) або модульна контрольна з кодомid_mod (якщоid_mod! = Null). Якщо включений – генерується повідомлення про помилку, інакше в таблицю ContentModulWP додається запис із зазначеними атрибутами.

Процедура ContentModulWPEdit оновлює записи в таблиці ContentModulWP:

create procedure ContentModulWPEdit
@id_modWP int,
@id_test_old smallint,

@id_mod_old smallint,
@id_test_new smallint,
@id_mod_new smallint,
@per_cent smallint,
@type bit,
@k smallint output,
@msg varchar (100) output

Параметри процедури:

@id_modWP – код модуля робочої програми;
@id_test_old – код колишнього код комп'ютерного тесту;
@id_mod_old – код колишнього змістовного модуля;
@id_test_new – код нового комп'ютерного тесту;
@id_mod_new – код нового змістовного модуля;
@per_cent – % від бали за модуль робочої програми;
@type – комп'ютерний тест чи ні;
@k – код помилки, вихідний параметр;
@msg – вихідний параметр, повідомлення про помилку.

Для змістовного модуля робочої програми, код якого дорівнює id_modWP, процедура ConentModulWPEdit привласнює нові значення атрибутів per_cent, type. Якщо змінюється в змістовному модулі назву і модуль з такою назвою вже є в робочій програмі, генерується повідомлення про помилку.

Процедура WorkProgAdd додає записи в таблиці WorkProg:

create procedure WorkProgAdd
@id_wp int,
@name_wp varchar(60),
@id_sub smallint,
@id_plan smallint,
@k smallint output,
@msg varchar (100) output

Параметри процедури:

@id_wp – код робочої програми;
@name_wp – назва робочої програми;
@id_sub – код дисципліни;
@id_plan – код навчального плану;
@k – код помилки, вихідний параметр;
@msg – повідомлення про помилку, вихідний параметр.

Процедура WorkProgAdd перевіряє, чи є вже в БД запис робочої програми з такою назвою. Якщо такий запис є, то генерується повідомлення про помилку. Якщо записи з вказаною назвою робочої програми немає, то в таблицю WorkProgAdd додається запис із зазначеними значеннями атрибутів.

Процедура WorkProgdel видаляє запис з таблиці WorkProg:

Процедура WorkProgEdit оновлює записи в таблиці WorkProg:

```
create  procedure WorkProgEdit
  @id_wp int,
  @name_wp varchar (60),
  @k smallint output,
  @msg varchar (100) output
```

Параметри процедури:

@id_wp – код робочої програми;;

@name_wp – назва робочої програми;

@k – код помилки, вихідний параметр;

@msg – повідомлення про помилку, вихідний параметр.

Для робочої програми, код якого дорівнюєid_WP, процедура WorkProgEdit привласнює нові значення атрибуту name_wp. Якщо змінюється в змістовному модулі назву і модуль з такою назвою вже є в робочій програмі, генерується повідомлення про помилку.

6 ОПИС РОЗРОБЛЕНОГО КЛІЄНТСЬКОГО ДОДАТКУ

6.1 Загальні відомості

Програма розроблена в середовищі розробки додатків Visual Studio 2010, може працювати під управлінням операційної системи Windows NT / 2000 / XP / Vista / 7 / 8 і будь-який інший сумісної мережевою операційною системою. Програма працює як додаток до бази даних, тому в локальній мережі в якій працює програма має бути сервер баз даних із завантаженою базою «Університет».

6.2 Функціональне призначення

Програма PavlenkoForms розроблена як додаток до бази даних «Університет». Програма призначена для перегляду даних (навчальні плани, контингент студентів, навчальні дисципліни, навчальне навантаження кафедри, навантаження викладачів і т.д.) в зручному вигляді.

Програма також дозволяє зручно змінювати ті дані, які має право оновлювати працівник кафедри, що відповідає за розподіл навчального навантаження між викладачами.

6.3 Опис програми

6.3.1 Керівництво програміста

Програма написана у вигляді додатку PavlenkoForms, складається з форми Form1, яка є об'єктом відповідного класу.

Клас основної форми:

public partial class Form1: Form

Конструктор класу Form1:

public Form1()

Цей конструктор без параметрів, він ініціалізує змінні типу команд, за допомогою яких здійснюється виклик збережених процедур.

Клас:

public partial class List_Item: Object

Клас List_Item призначений для роботи зі списками. У класі List_Item є методи по додаванню, видалення, вставці елементів.

Методи класу форми Form1.

Методи ініціалізації збережених процедур:

private void init_commands_AssignLector()

Служить для ініціалізації команди виклику збереженої процедури AssignLectorCmd.

private void init_commands_RemoveLector()

Служить для ініціалізації команди виклику збереженої процедури RemoveLectorCmd.

private void init_commands_AssingTeacher()

Служить для ініціалізації команди виклику збереженої процедури AssingTeacherCmd.

private void init_commands_RemoveTeacher()

Служить для ініціалізації команди виклику збереженої процедури RemoveTeacherCmd.

Решті методи класу Form1 являються обробниками подій натискання на кнопки або вибору пункту випадаючого списку.

Обробник події натискання на кнопку AssignLectorButt:

private void AssignLectorButt _Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку AssignLectorButt (призначити лектора).

Обробник події натискання на кнопку RemoveLectorButt:

private void RemoveLectorButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку RemoveLectorButt (зняти лектора).

Обробник події натискання на кнопку AssingTeacherButt:

private void AssingTeacherButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку AssingTeacherButt (призачити викладача).

Обробник події натискання на кнопку RemoveTeacherButt:

private void RemoveTeacherButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку RemoveTeacherButt (зняти викладача).

Обробник події натискання на кнопку AssingFlowButt:

private void AssingFlowButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку AssingFlowButt (призачити потік).

Обробник події натискання на кнопку RemoveFlowButt:

private void RemoveFlowButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку RemoveFlowButt (зняти потік).

Обробник події натискання на кнопку DivideFlowButt:

private void DivivdeFlowButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку DivideFlowButt (розділити потік).

Обробник події натискання на кнопку RemoveDivideFlowButt:

private void RemoveDivivdeFlowButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку RemoveDivideFlowButt (видалити розділення потоку).

Обробник події натискання на кнопку AddKindControlSubjectButt:

private void AddKindControlSubjectButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку AddKindControlSubjectButt (додати вид контролю з дисципліни).

Обробник події натискання на кнопку EditKindControlSubjectButt:

private void EditKindControlSubjectButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку EditKindControlSubjectButt (змінити вид контролю з дисципліни).

Обробник події натискання на кнопку DelKindControlSubjectButt:

private void DelKindControlSubjectButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку DelKindControlSubjectButt (видалити вид контролю з дисципліни).

Обробник події натискання на кнопку AddDiplomProjectButt:

private void AddDiplomProjectButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку AddDiplomProjectButt (додати дипломний проект).

Обробник події натискання на кнопку EditDiplomProjectButt:

private void EditDiplomProjectButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку EditDiplomProjectButt (змінити дипломний проект).

Обробник події натискання на кнопку DelDiplomProjectButt:

private void DelDiplomProjectButt _Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку DelDiplomProjectButt (видалити дипломний проект).

Обробник події вибору пунктів випадаючого списку FacultetcomboBox:

private void FacultetcomboBox_SelectedIndexChanged (object sender, EventArgs e)

Цей метод змінної id_f привласнює значення ідентифікатора факультету відповідного пункту випадаючого списку. Для обраного факультету на форму виводяться: назва групи, кількість студентів.

Обробник події вибору пунктів випадаючого списку LectorcomboBox:

private void LectorcomboBox_SelectedIndexChanged (object sender, EventArgs e)

Цей метод змінної id_dep привласнює значення ідентифікатора кафедри відповідного пункту випадаючого списку. Для обраного лектора на форму виводяться: дисципліна, група, потік, призвище та ім'я по батькові.

Обробник події вибору пунктів випадаючого списку TeachercomboBox:

private void TeachercomboBox_SelectedIndexChanged (object sender, EventArgs e)

Цей метод змінної id_dep привласнює значення ідентифікатора кафедри відповідного пункту випадаючого списку. Для обраного викладача на форму виводяться: : дисципліна, група, потік, призвище та ім'я по батькові.

private void OKLector_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку OKLector.

private void OKTeacher_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку OKTeacher.

private void CancelLector_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку CancelLector.

private void CancelTeacher_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку CancelTeacher.

private void OKYearTeacher_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку OKYearTeacher.

private void OKYearCafedra_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку OKYearCafedra.

Обробник події переходу на нову строку dataGridView12_RowEnter:

private void dataGridView12_RowEnter (object sender, dataGridViewCellEventArgs e)

Цей метод дозволяє перемикати данні в іншій таблиці у той момент, коли користувач обирає для перегляду іншу строку.

6.3.2 Посібник користувача.

Програма запускається як звичайний exe-файл, на екрані з'являється головна форма програм. На головній формі знаходиться багатосторінковий компонент з чотирма вкладками.

На першій вкладці можна переглянути навчальні плани, як показано на рис. 6.1.

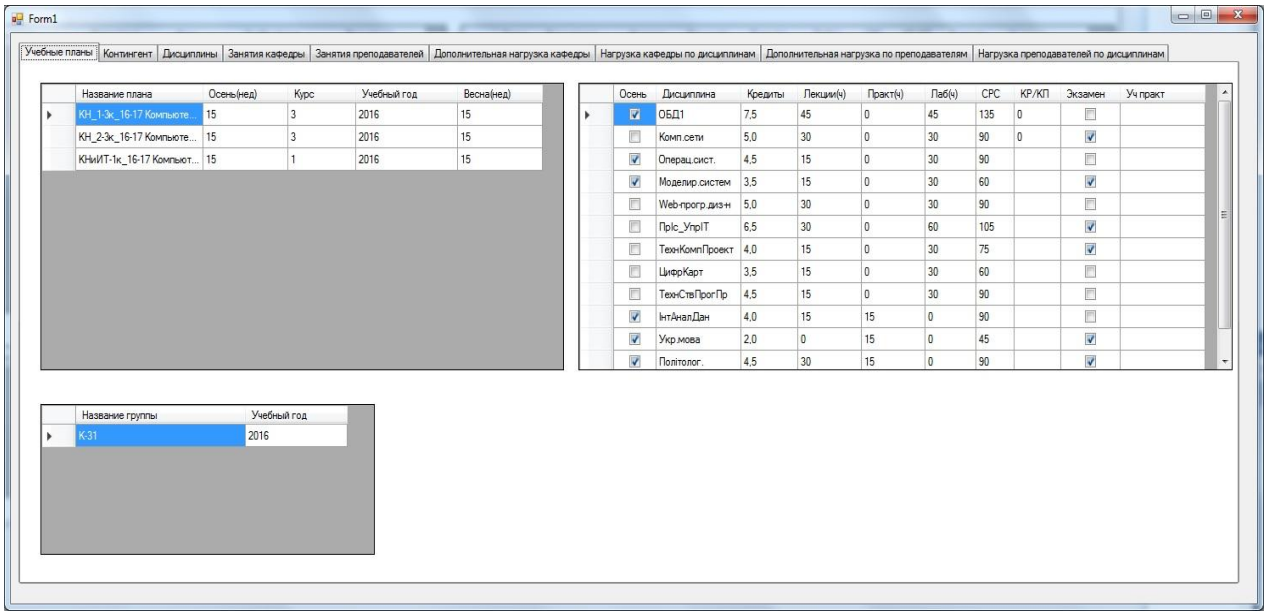


Рисунок 6.1 – Форма для перегляду навчальних планів

Як видно з рис. 6.1, програма дозволяє переглянути наявність навчальних планів для вибраного навчального року в табличному компоненті зверху форми ліворуч.

Для поточного навчального плану в табличному компоненті праворуч можна переглянути його наповнення, тобто перелік дисциплін, що вивчаються у кожному семестрі, з вказівкою для кожної дисципліни кількості кредитів, кількості годин лекцій, практичних, лабораторних занять, СРС, наявність КР/КП, екзамену та навчальні практики. У табличному компоненті понизу форми виводиться перелік академічних груп, які навчаються в вибраному навчальному році за поточним навчальним планом.

На другій вкладці можна переглянути контингент студентів (рис. 6.2.). Доки користувач не обере факультет, він нічого не побачить.

На рису. 6.3 зображені групи факультету комп'ютерних наук.

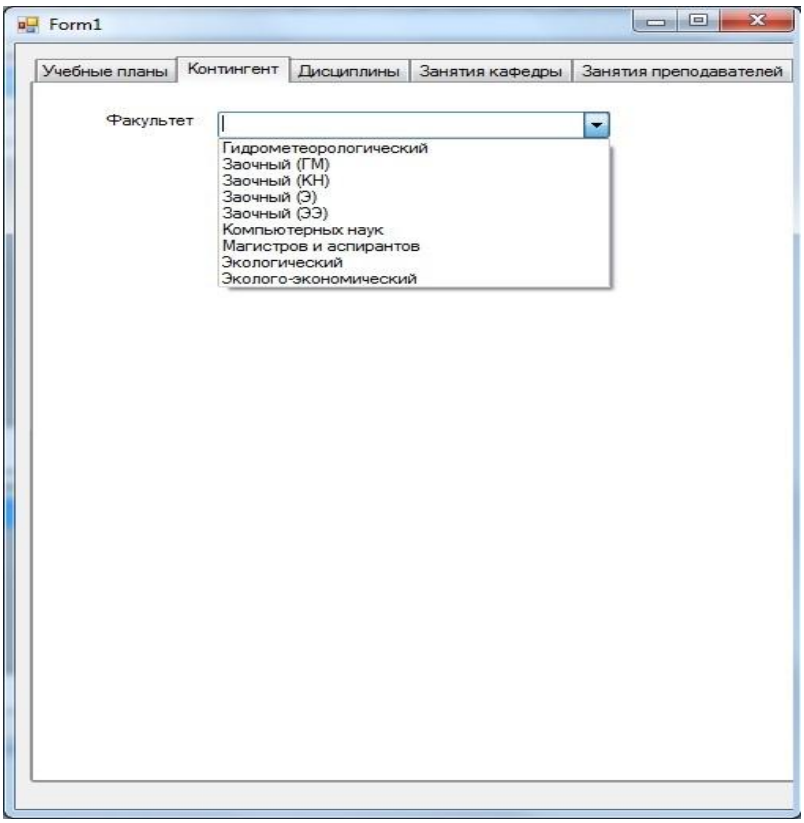


Рисунок 6.2 – Вибір факультету для перегляду контингенту студентів

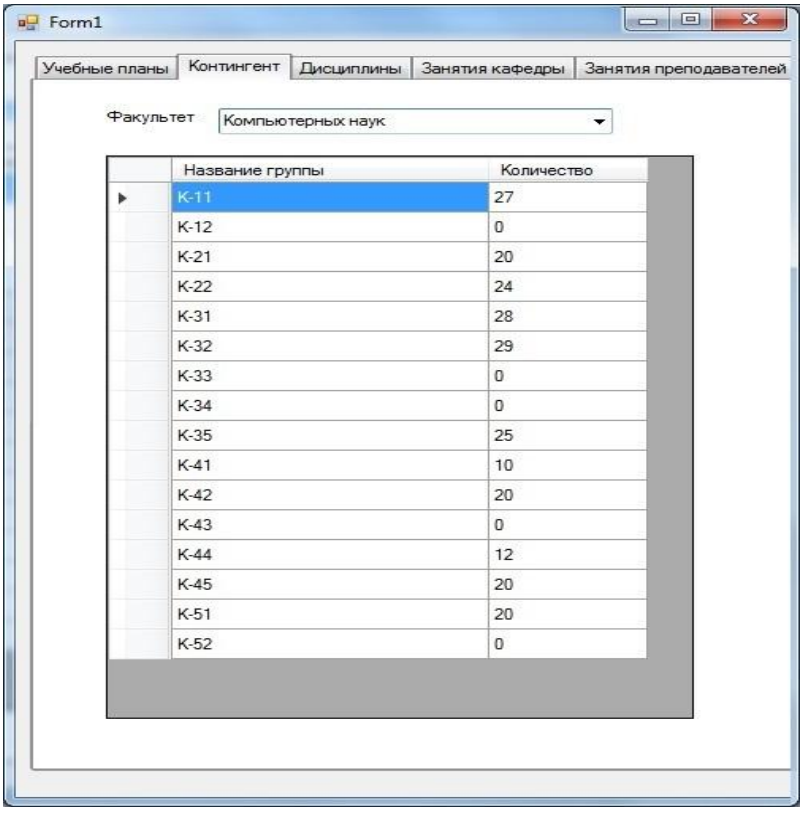


Рисунок 6.3 – Перегляд контингенту студентів вибраного факультету

На вкладці «Дисциплины» можна побачити перелік навчальних дисциплін кафедри (рис. 6.4).

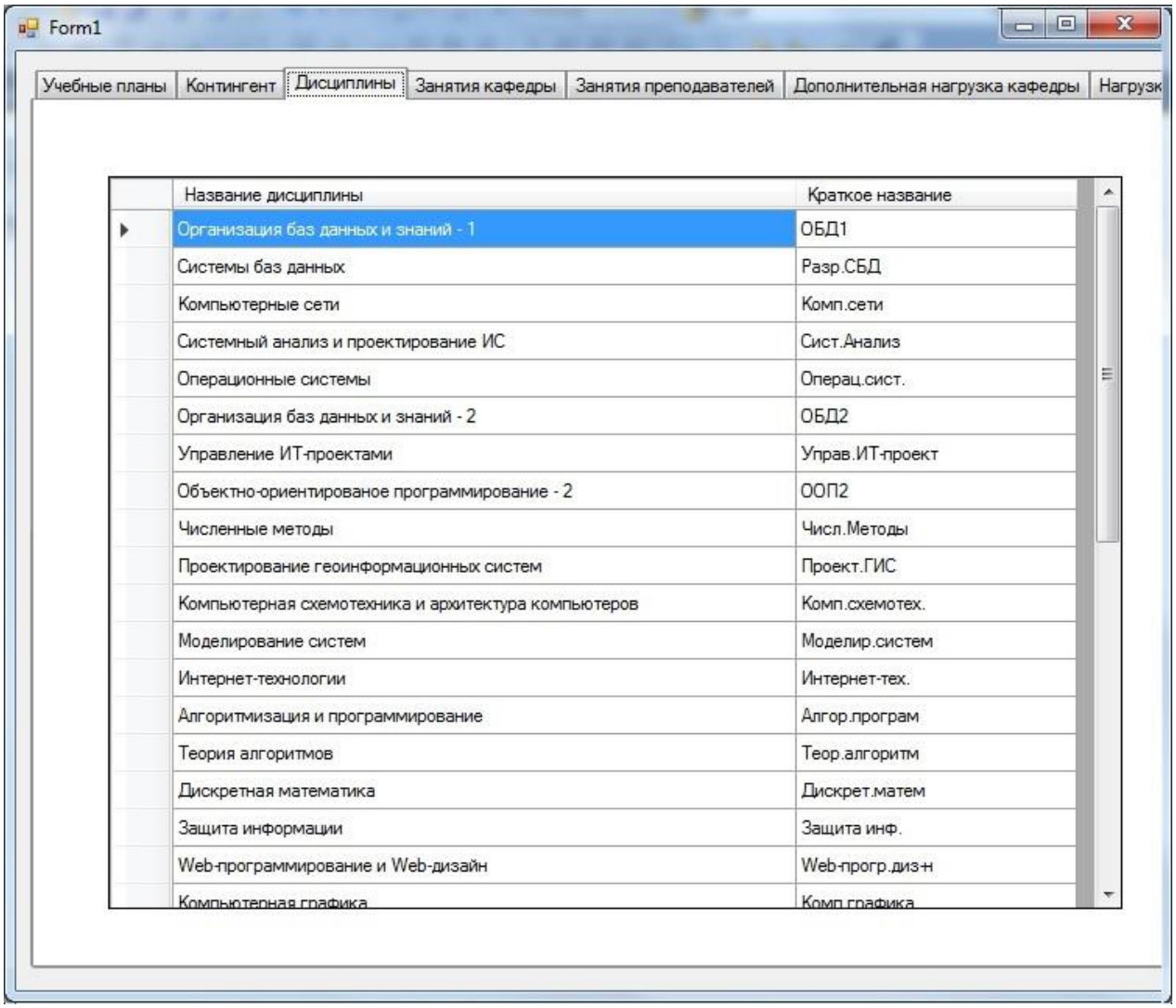


Рисунок 6.4 – Форма для перегляду дисциплін

Користувач «Співробітник кафедри» може лише переглядати список дисциплін, але не оновлювати. Якщо цей перелік не відповідає списку тих дисциплін, що буде вести кафедра у розрахунковому році, потрібно звернутися до методичного відділу, щоб вони внесли зміни у закріплення дисциплін за кафедрами.

На рис. 6.5 зображена вкладка «Занятия кафедры». Потрібно вибрати навчальний рік, щоб побачити заняття за дисциплінами кафедри на вказаний рік.

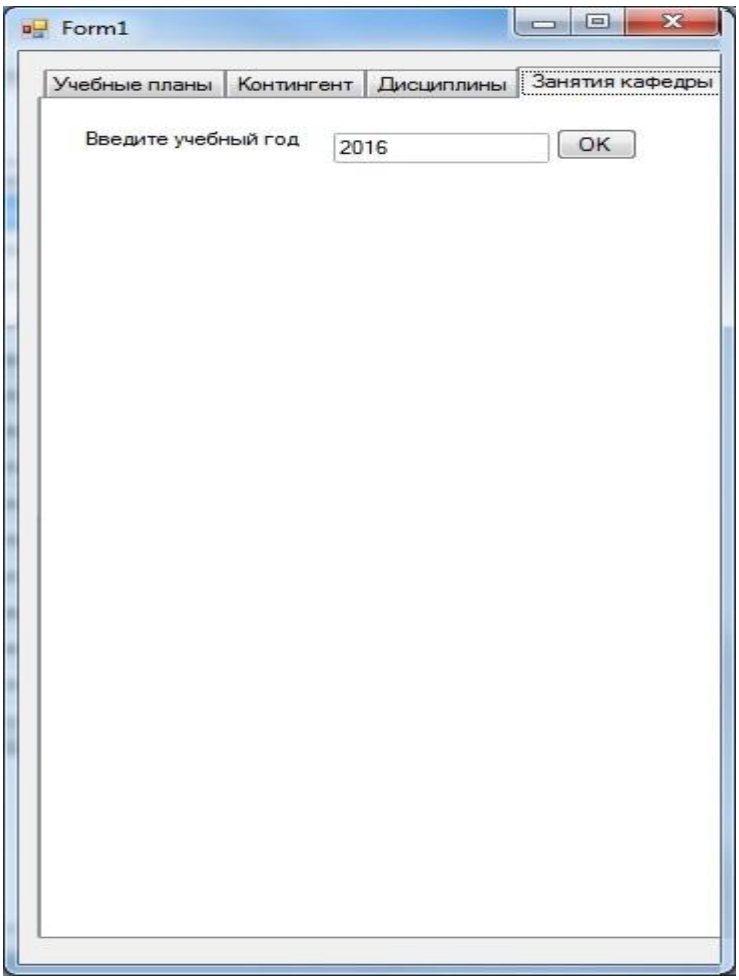


Рисунок 6.5 – Вибір навчального року для перегляду запланованих на рік занять кафедри

Коли викладач уввів навчальний рік, він побаче заняття, що заплановані за навчальними планами для академічних груп з дисциплін кафедри (рис 6.6).

Навантаження згідно навчальних планів розраховується в базі даних автоматично для кожної дисципліни для кожної групи. Але кафедра може дещо змінити це навантаження, якщо об’єднає деякі групи для проведення лекційних занять, або розділить групи на підгрупи для проведення лабораторних занять або навчальних практик.

Для виконання вказаних дій на формі присутні 4 кнопки (рис.6.6). Якщо поточним заняття у списку є лекція, для якої не призначений потік, можна натиснути на кнопку «Назначить поток» та вибрати потік.

Якщо для деяких груп вже призначений потік для читання лекцій, можна для лекційного заняття будь-якої групи цього потоку натиснути кнопку

«Убрать поток». Після цього для всіх груп цього потоку лекції з цієї дисципліни будуть читатись окремо, якщо групи на об’єднати в інші потоки.

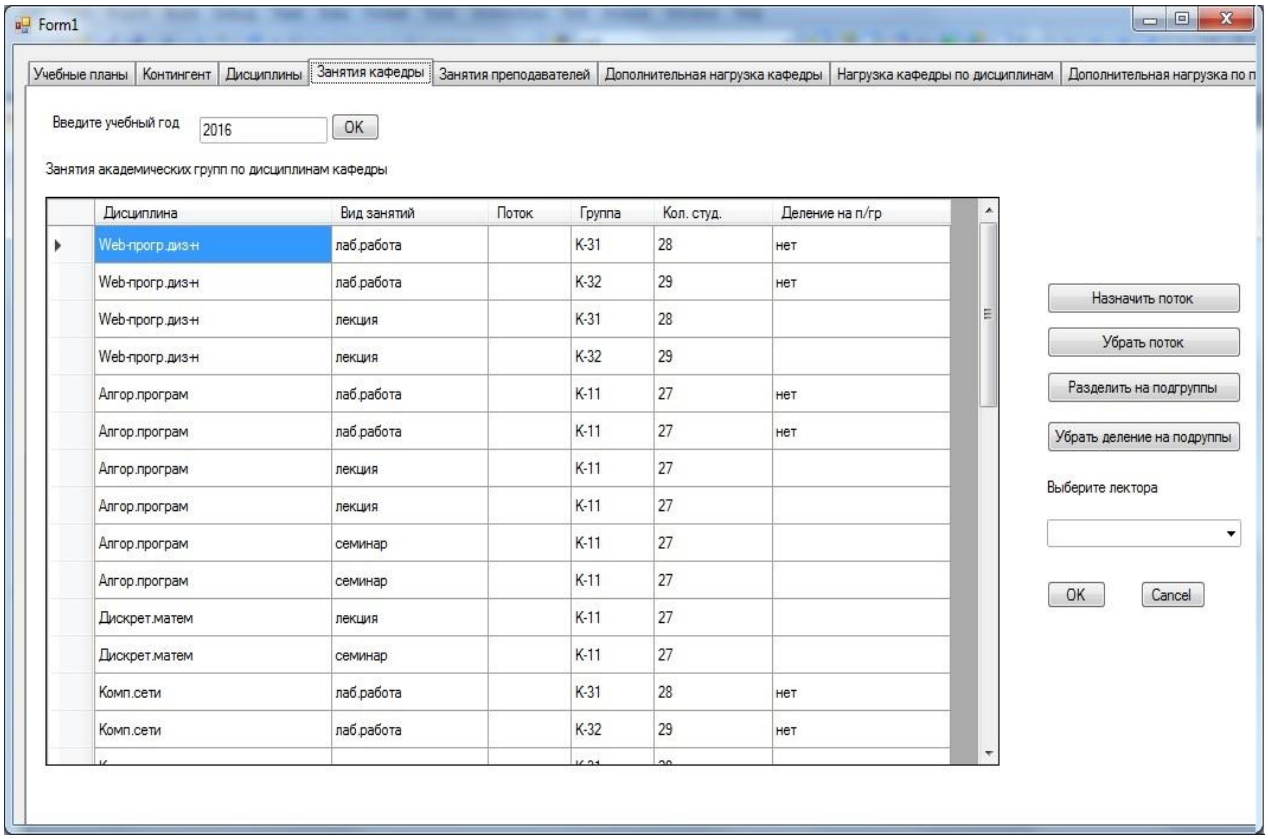


Рисунок 6.6 – Заповнена форма для перегляду занять кафедри

Для лабораторних занять та навчальних практик можна розподіляти групи на підгрупи, або відміняти назначений розподіл.

Для всіх занять кафедри необхідно призначити викладачів, які будуть їх вести. Для цього призначена вкладка «Занятия преподавателей». Для перегляду занять викладачів також потрібно задати навчальний рік (рис.6.7).

Після завдання навчального року виводиться перелік всіх занять по всіх дисциплінах кафедри на заданий навчальний рік (рис.6.8).

Заняття виводиться у два табличних компоненти: ліворуч виводиться список лекцій, праворуч – список семінаров, лабораторних занять та навчальних практик. Коли для лекції з дисципліни заданий потік, тоді у таблиці ліворуч виводиться один рядок для об’єднаної лекції всіх груп потоку.

Для кожного заняття потрібно призначити викладача. Для цього на формі є кнопки «Назначить лектора» та «Назначить преподавателя». Для зміни викладача потрібно спочатку зняти з заняття вже призначеного викладача кнопками «Снять лектора» та «Снять преподавателя».

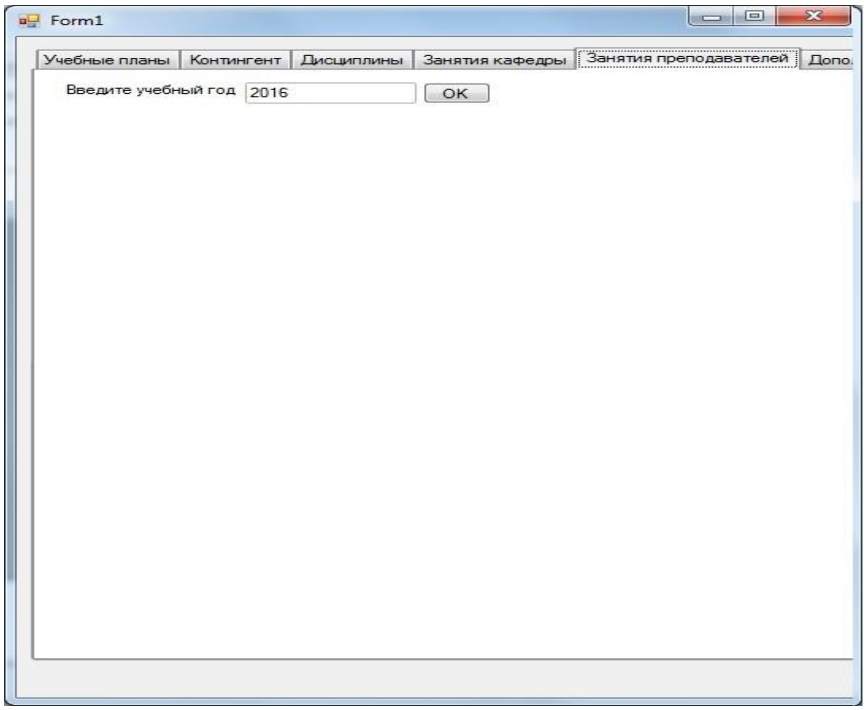


Рисунок 6.7 – Форма для перегляду занять викладачів

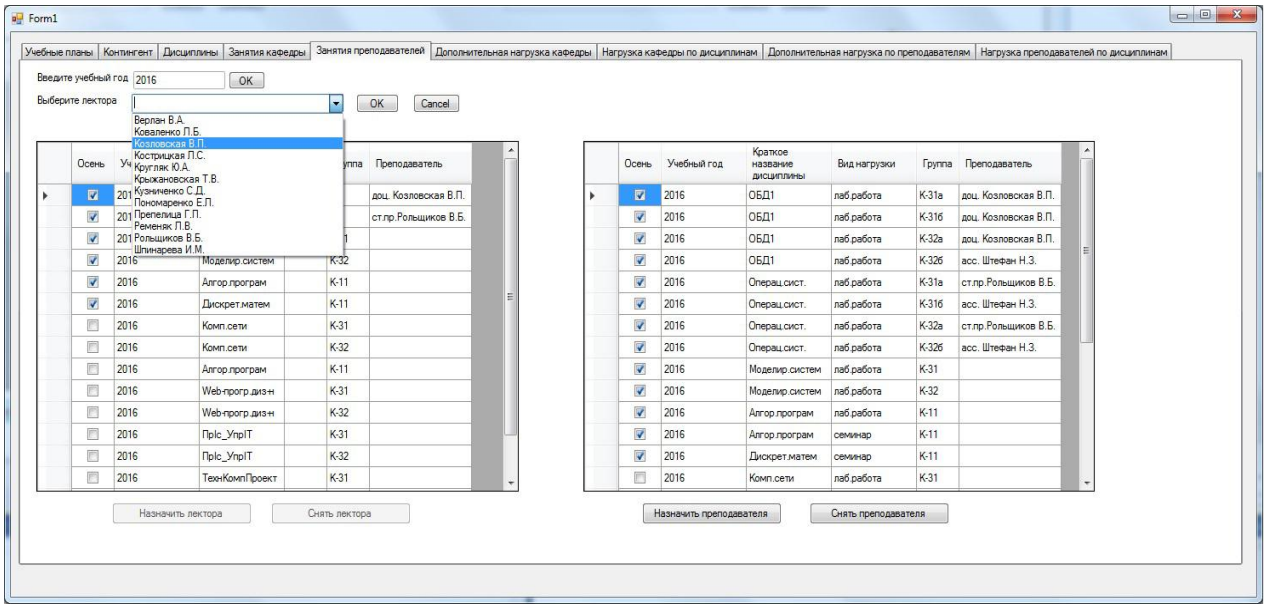


Рисунок 6.8 – Список викладачів кафедри для призначення на лекційні заняття

При натисканні кнопок «Назначить лектора» та «Назначить преподавателя» відкривається випадний список з переліком викладачів, з якого можна вибрати потрібного викладача (рис. 6.9).

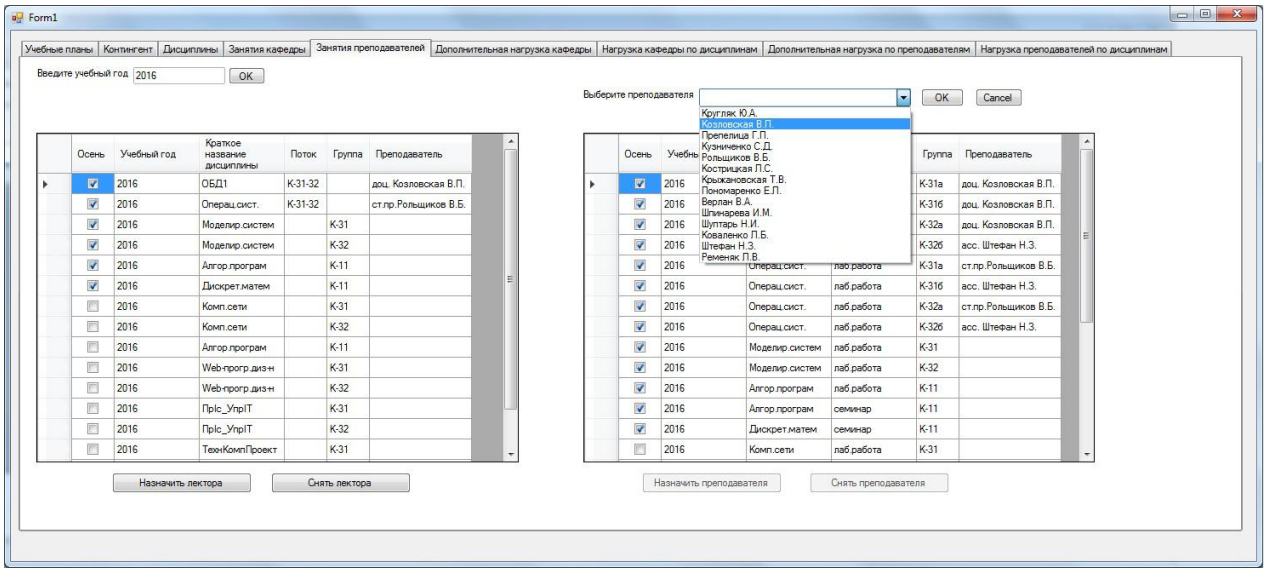


Рисунок 6.9 – Список викладачів кафедри для призначення на практичні заняття

Крім аудиторного навантаження з дисципліни існує навантаження, обумовлене контролем самостійної роботи студентів (СРС). Для кожної дисципліни потрібно задати види контролю СРС та їх кількість, як показано на рис. 6.10. Кнопки «Добавить», «Изменить», «Удалить» дозволяють змінити перелік та кількість СРС для кожної дисципліни.

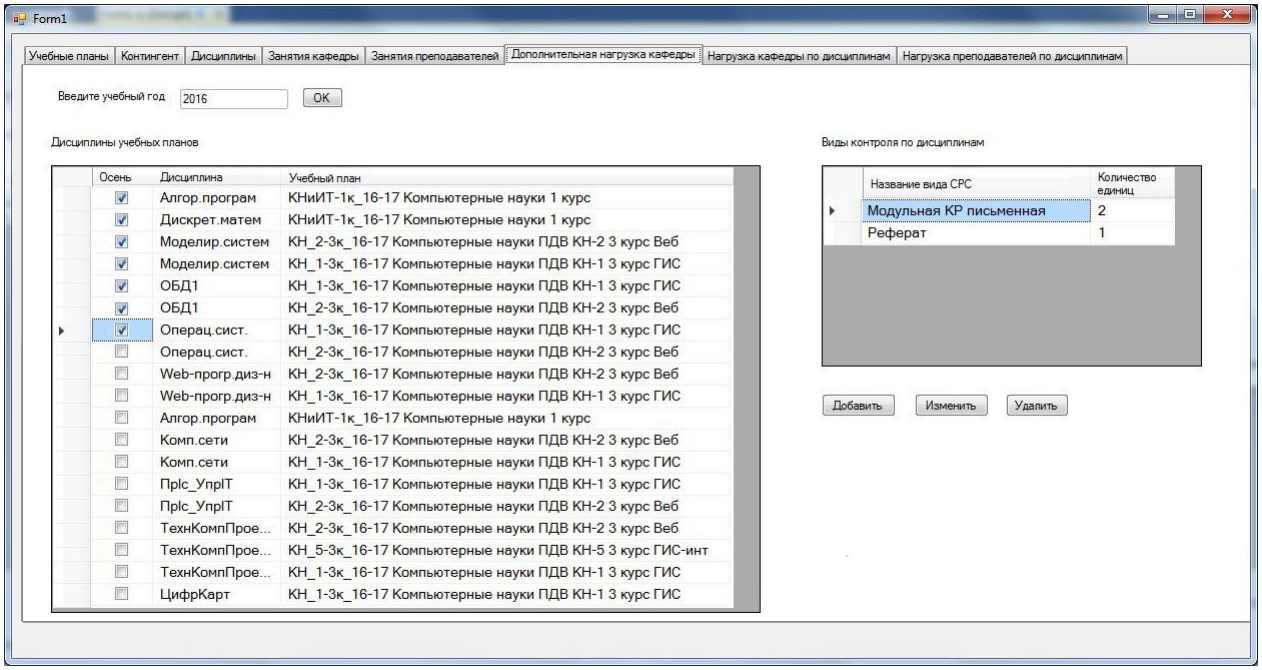
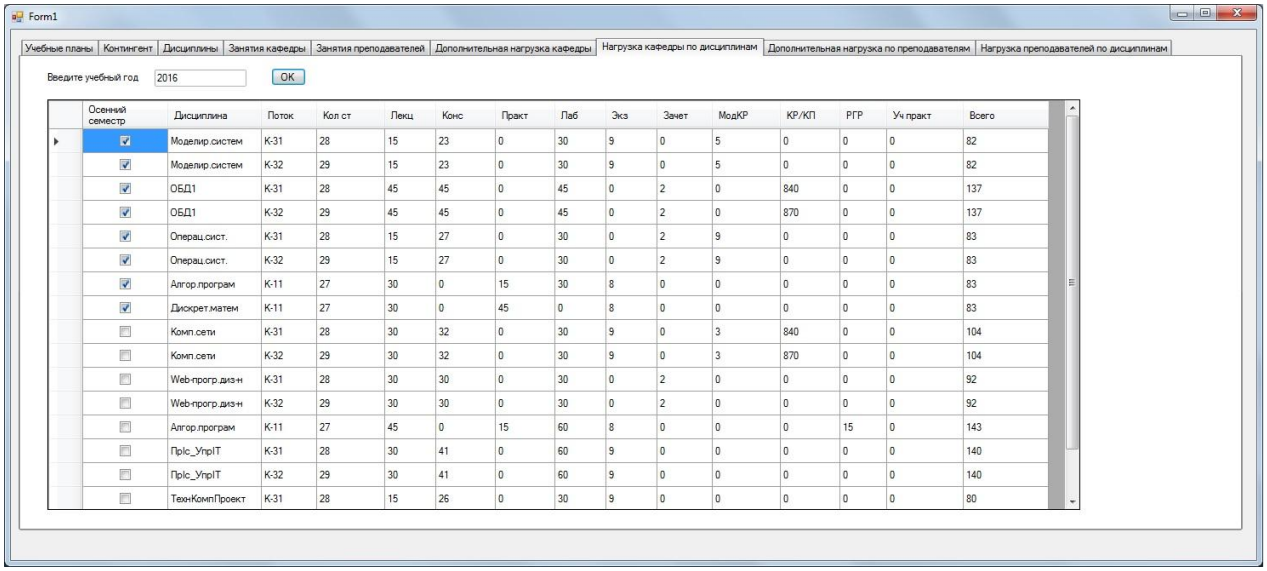


Рисунок 6.10 – Форма для перегляду додаткового навантаження кафедри

Навантаження по дисциплінах кафедри розраховується автоматично. На вкладці «Нагрузка кафедры по дисциплинам» можна переглянути це навантаження (рис. 6.11).

На вкладці «Нагрузка преподавателей по дисциплинам» можна переглянути навантаження окремого викладача кафедри. Для цього спочатку потрібно вибрати викладача зі списку (рис. 6.12).



Осенний семестр	Дисциплина	Поток	Кол ст	Лекц	Конс	Практ	Лаб	Экз	Зачет	МодКР	КР/КП	РГР	Уч практ	Всего
☑	Моделир.систем	К-31	28	15	23	0	30	9	0	5	0	0	0	82
☑	Моделир.систем	К-32	29	15	23	0	30	9	0	5	0	0	0	82
☑	ОБД1	К-31	28	45	45	0	45	0	2	0	840	0	0	137
☑	ОБД1	К-32	29	45	45	0	45	0	2	0	870	0	0	137
☑	Операц.сист.	К-31	28	15	27	0	30	0	2	9	0	0	0	83
☑	Операц.сист.	К-32	29	15	27	0	30	0	2	9	0	0	0	83
☑	Алгор.програм	К-11	27	30	0	15	30	8	0	0	0	0	0	83
☑	Дискрет.матем	К-11	27	30	0	45	0	8	0	0	0	0	0	83
☐	Комп.сети	К-31	28	30	32	0	30	9	0	3	840	0	0	104
☐	Комп.сети	К-32	29	30	32	0	30	9	0	3	870	0	0	104
☐	Web-прогр.диз-н	К-31	28	30	30	0	30	0	2	0	0	0	0	92
☐	Web-прогр.диз-н	К-32	29	30	30	0	30	0	2	0	0	0	0	92
☐	Алгор.програм	К-11	27	45	0	15	60	8	0	0	0	15	0	143
☐	ПрІс_УпрІТ	К-31	28	30	41	0	60	9	0	0	0	0	0	140
☐	ПрІс_УпрІТ	К-32	29	30	41	0	60	9	0	0	0	0	0	140
☐	ТехнКомпПроект	К-31	28	15	26	0	30	9	0	0	0	0	0	80

Рисунок 6.11 – Заповнена форма для перегляду навантаження кафедри з дисциплін

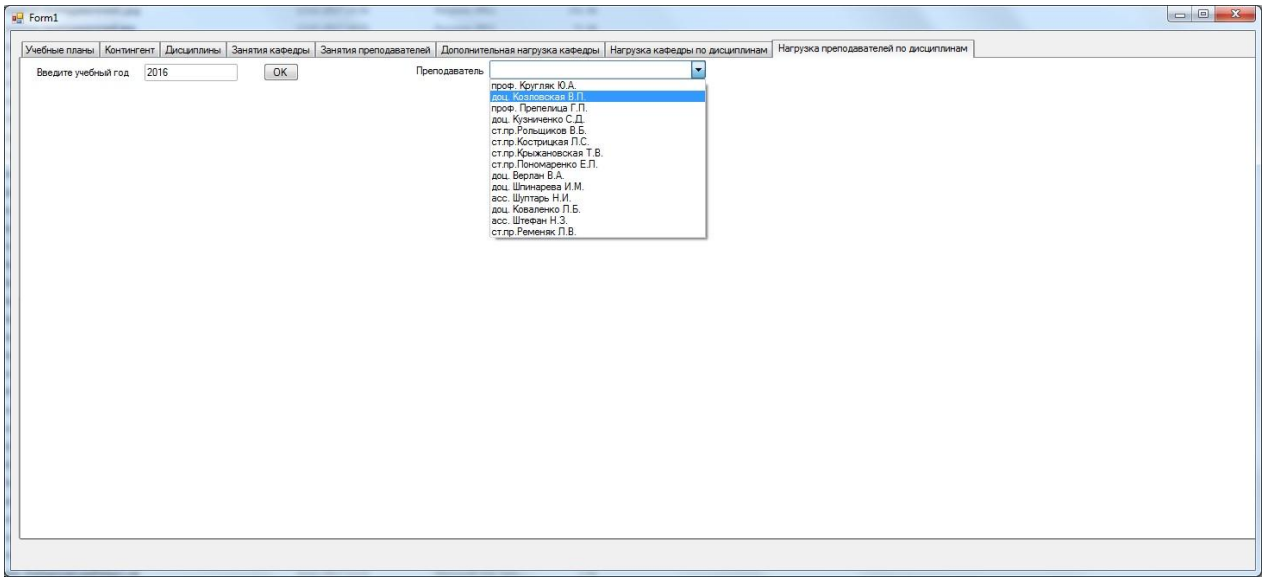


Рисунок 6.12 – Вибір викладача для перегляду його навантаження

Для вибраного викладача на формі з’являєть перелік дисциплін, які він буде вести у розрахунковому році, та розмір навантаження по всіх видах для кожної дисципліни (рис. 6.13).

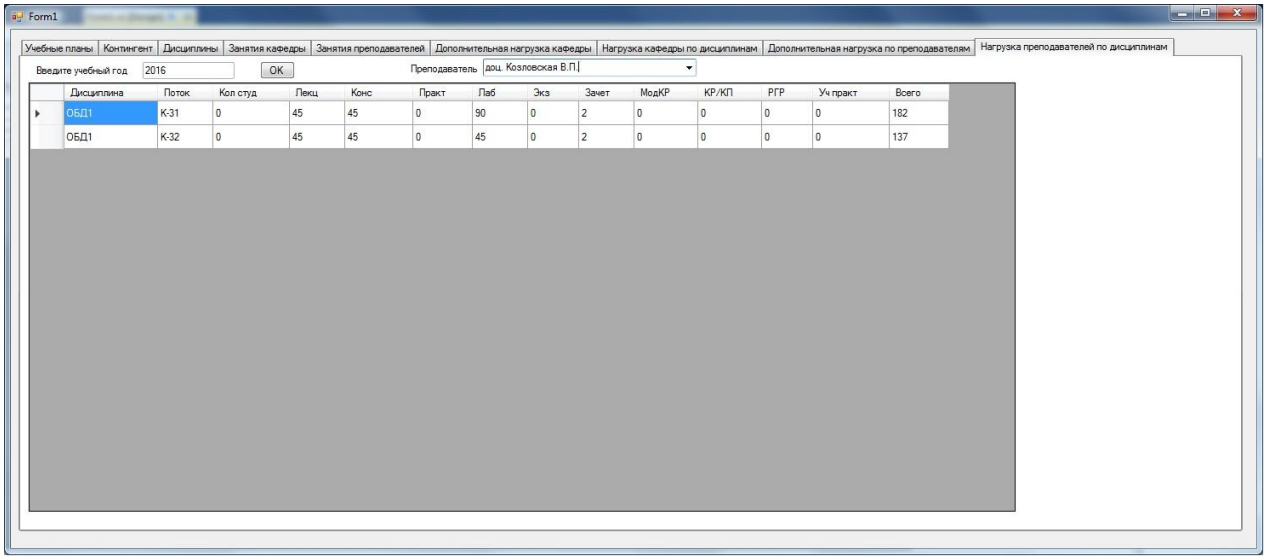


Рисунок 6.13 – Заповнена форма для перегляду навантаження викладачів з дисциплін

Розроблена програма дозволяє дуже просто розподіляти навчальне навантаження між викладами кафедри. Також вона дозволяє уникнути розрахунків загального навчального навантаження кафедри та навантаження кожного окремого викладача, тому що ці розрахунки проводяться автоматично.

ВИСНОВКИ

Метою комплексної магістерської роботи було інтегрування у єдину інформаційну систему «Університет» прикладних систем баз даних «Електронний деканат», «Інтегральні відомості», «Тестуюча система», «Навантаження викладача», «Розклад занять».

Задачею даної частини комплексної магістерської роботи було було інтегрування у ІС «Університет» прикладних систем баз даних «Навантаження викладача» та «Розклад занять», а також створення програмного застосування у вигляді офісного додатку «АРМ співробітника кафедри».

Особисто автором даної частини комплексної магістерської роботи були отримані результати:

- виконаний системний аналіз предметної області з метою оцінки повноти концептуальної моделі для вирішення поставленої задачі;
- доповнена концептуальна модель новими сутностями для підсхеми «Навантаження викладача та розклад занять»;
- виконане логічне і фізичне проектування бази даних у відповідності до зміненої концептуальної моделі бази даних;
- розроблене програмне застосування «АРМ співробітника кафедри».

Розроблена інформаційна система «Університет» позволяє автоматизувати всі стадії навчального процесу в університеті. Розроблене програмне застосування дозволяє дуже просто розподіляти навчальне навантаження між викладами кафедри. Також вона дозволяє уникнути розрахунків загального навчального навантаження кафедри та навантаження кожного окремого викладача, тому що ці розрахунки проводяться автоматично.

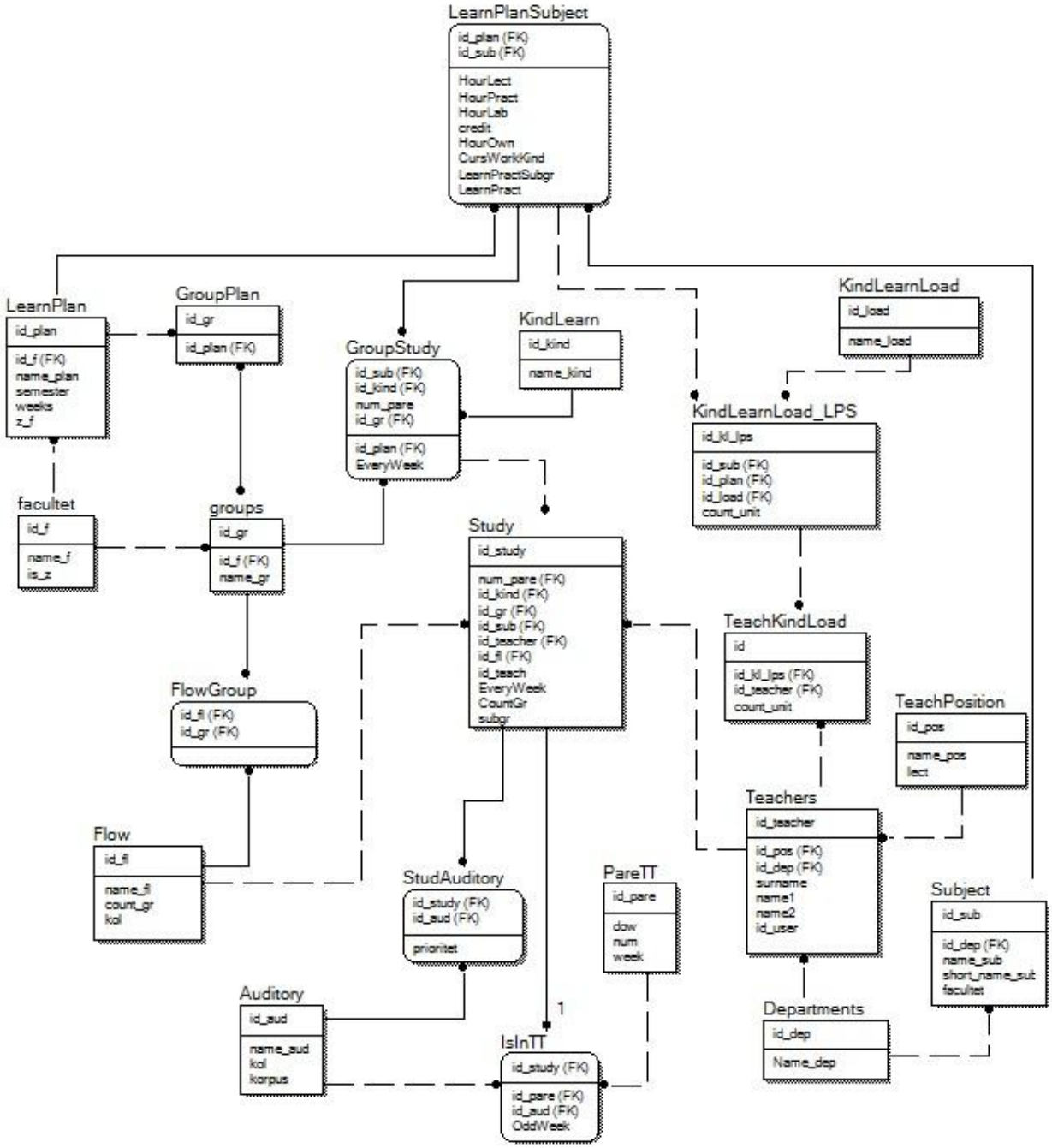
Для повноцінної роботи ІС «Університет» в подальшому потрібно розробити програмного застосування для інших груп користувачів: співробітників методичного відділу, співробітників навчального відділу, викладачів.

ПЕРЕЛІК ПОСИЛАНЬ

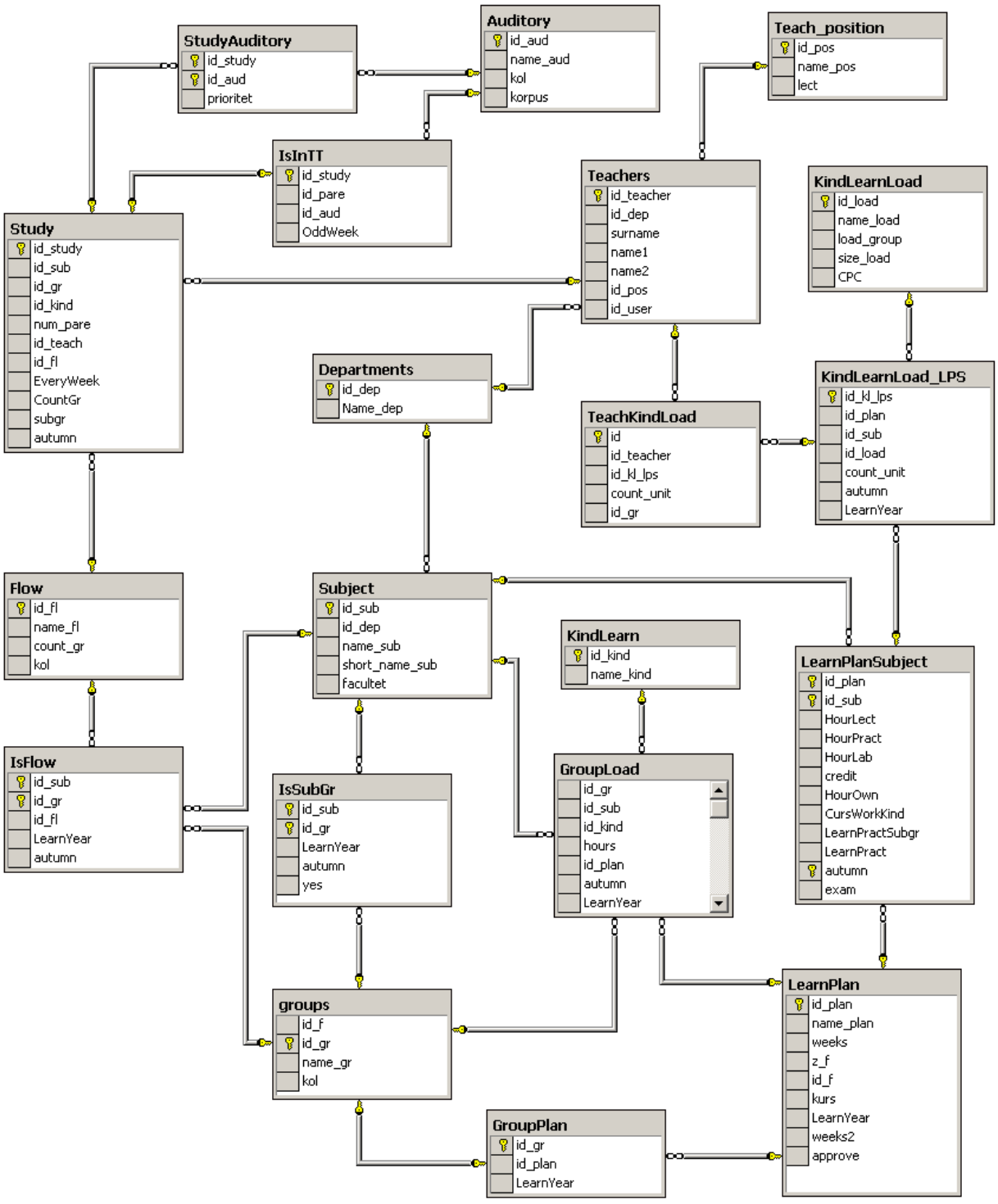
1. Албахари Джозеф, Албахари Бен. C# 5.0. Справочник. Полное описание языка.: Пер. с англ. – Ю. Артеменко. – М.: Издательский дом «Вильямс», 2014. – 1008 с.
2. Вільям Р. Станек. Microsoft SQL Server 2008. Справочник администратора. – СПб.: БХВ-Петербург, 2013. – 720 с.
3. Душан Петкович. Microsoft SQL Server 2008. Руководство для начинающих. – СПб.: БХВ-Петербург, 2009. – 752 с.
4. Дейт К. Дж.. Введение в системы баз данных, 6-е издание.: Пер. с англ. – К.; М.; СПб: Издательский дом «Вильямс», 2000 – 848 с.
5. Джеми Макленнен, Чжаохуей Танг, Криват Богдан. Microsoft SQL Server 2008. Data Mining – интелектуальный анализ даннных.: Пер. с англ. – А. Лашкевич. – СПб.: БХВ-Петербург, 2009. – 720 с.
6. Ицик Бен-Ган. Microsoft SQL Server 2008. Основы T-SQL. – СПб. : БХВ-Петербург, 2009. – 430 с.
7. Леонард Лобел, Эндрю Дж. Браст, Стивен Форте. Разработка приложений на основе Microsoft SQL Server 2008. – СПб. : БХВ-Петербург, Русская Редакция, 2010. – 1024 с.
8. Мамаев Е.В. Microsoft SQL Server 2000. – СПб. : БХВ-Петербург, 2004. – 1260 с.
9. Новиков Б.А. , Г. Р. Домбровская. Настройка приложений баз данных. – СПб.: БХВ-Петербург, 2006. – 729 с.
10. Никита Культин. Основы программирования в Microsoft Visual C# 2010. – СПб.: БХВ-Петербург, 2011. – 368 с.
11. Рудикова Л. В.. Базы данных. Разработка приложений. – СПб.: БХВ-Петербург, 2003 – 1440 с.
12. Робин Дьюсон. SQL Server 2008 для начинающих разработчиков. – СПб.: БХВ-Петербург, 2009. – 704 с.
13. Томас Конноли, Каролин Бегг. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание.: Пер. с англ. – М. : Издательский дом «Вильямс», 2003 – 1440 с.
14. Эндрю Стиллмен, Дженнифер Грин. Изучаем C #. 3-е издание. – СПб. : Питер, 2014. – 816 с.
15. Энтони Молинаро. SQL. Сборник рецептов.: Пер. с англ. – А. Лашкевич. – СПб.: Символ-Плюс, 2009. – 720 с.

Д О Д А Т К И

ДОДАТОК А – ER-ДІАГРАМА БД «УНІВЕРСИТЕТ»: ПІДСХЕМА «НАВАНТАЖЕННЯ ВИКЛАДАЧА ТА РОЗКЛАД ЗАНЯТЬ»



ДОДАТОК Б – ЛОГІЧНА МОДЕЛЬ БД «УНІВЕРСИТЕТ»: ПІДСХЕМА «НАВАНТАЖЕННЯ ВИКЛАДАЧА ТА РОЗКЛАД ЗАНЯТЬ»



ДОДАТОК В – ФІЗИЧНА СХЕМА БАЗИ ДАНИХ

```
CREATE TABLE [KindLearnLoad] (
      [id_load] [smallint] IDENTITY (1, 1) NOT NULL PRIMARY KEY  ,
      [name_load] [varchar] (50) NOT NULL UNIQUE  ,
      [load_group] [bit] NULL ,
      [size_load] [numeric](4, 2) NOT NULL ,
      [CPC] [bit] NULL ,
GO

CREATE TABLE [KindLearnLoad_LPS] (
      [id_kl_lps] [int] IDENTITY (1, 1) NOT NULL PRIMARY KEY  ,
      [id_plan] [smallint] NOT NULL ,
      [id_sub] [smallint] NOT NULL ,
      [id_load] [smallint] NULL ,
      [count_unit] [smallint] NOT NULL ,
      [autumn] [bit] NOT NULL ,
      [LearnYear] [smallint] NOT NULL ,
       UNIQUE  ([id_plan],[id_sub],[id_load])  ON [PRIMARY] ,
FOREIGN KEY ([id_load]) REFERENCES [KindLearnLoad] ([id_load]),
FOREIGN KEY ([id_plan],[id_sub],[autumn]) REFERENCES
[LearnPlanSubject] ([id_plan],[id_sub],[autumn]) ON DELETE CASCADE
) ON [PRIMARY]
GO

CREATE TABLE [TeachLoadSub] (
      [id_gl] [int] NULL ,
      [id_teach] [smallint] NULL ,
      [subgr] [tinyint] NULL ,
      [value] [numeric](3, 2) NULL DEFAULT (1.0),
       UNIQUE  ([id_gl],[subgr])  ON [PRIMARY] ,
       FOREIGN KEY ([id_gl]) REFERENCES [GroupLoad] ([id_gl]),
       FOREIGN KEY([id_teach]) REFERENCES [Teachers] ([id_teacher])
) ON [PRIMARY]
GO

CREATE TABLE [TeachKindLoad] (
      [id] [int] IDENTITY (1, 1) NOT NULL PRIMARY KEY  ,
      [id_teacher] [smallint] NULL ,
      [id_kl_lps] [int] NULL ,
      [count_unit] [smallint] NOT NULL ,
      [id_gr] [smallint] NULL ,
       UNIQUE  ([id_kl_lps],[id_teacher])  ON [PRIMARY] ,
FOREIGN KEY ([id_kl_lps])REFERENCES [KindLearnLoad_LPS]([id_kl_lps]),
FOREIGN KEY ([id_teacher]) REFERENCES [Teachers] ([id_teacher])
) ON [PRIMARY]
GO
```

ДОДАТОК Г – ПРЕДСТАВЛЕННЯ БАЗИ ДАНИХ

```
CREATE view IsFlowView as
select s.id_sub, gl.id_gr, ifl.id_fl, name_fl,
gl.LearnYear,gl.autumn,id_dep
from GroupLoad gl join IsFlow ifl on gl.id_gr=ifl.id_gr and
gl.id_sub=ifl.id_sub  and gl.LearnYear=ifl.LearnYear and
gl.autumn=ifl.autumn
join Flow f on f.id_fl=ifl.id_fl join Subject s on s.id_sub=gl.id_sub
where gl.id_kind=1

CREATE  view IsSubGrView as
     select s.id_sub, gl.id_gr,gl.LearnYear,gl.autumn,
     Sub_Gr=case when yes is null then 'нет' else 'да' end, id_dep
     from GroupLoad gl left outer join IsSubGr ifl on
gl.id_gr=ifl.id_gr and gl.id_sub=ifl.id_sub
                and gl.LearnYear=ifl.LearnYear and
gl.autumn=ifl.autumn  join Subject s on s.id_sub=gl.id_sub
     where gl.id_kind=3

CREATE view GroupLoadTeachView as
select v.*,id_gl,subgr
from TeachLoadSub tl left outer join  ViewTeachPos v on
v.id_teacher=tl.id_teach

Create view ShowLoadLector as
select distinct gl.id_sub,gl.autumn,gl.learnYear,short_name_sub,
id1=case when id_fl is null then gl.id_gr else id_fl end,
name_f=case when id_fl is null then '' else name_fl end,
name_g=case when id_fl is null then name_gr else '' end
,Prep=case when id_teacher is null then '' else Teach end,id_teacher,
s.id_dep
from Subject s join GroupLoad gl on s.id_sub=gl.id_sub join groups g
on g.id_gr=gl.id_gr
left outer join IsFlowView ifl
     on gl.id_sub=ifl.id_sub and gl.id_gr=ifl.id_gr and
gl.autumn=ifl.autumn and gl.learnYear=ifl.learnYear
left outer join GroupLoadTeachView v on gl.id_gl=v.id_gl
where id_kind=1

create view ShowLoadPract as
select  gl.id_sub,gl.autumn,gl.learnYear,short_name_sub,name_kind,
name_g=name_gr,Prep=case when id_teacher is null then '' else Teach
end,id_teacher,gl.id_gr, s.id_dep
from Subject s join GroupLoad gl on s.id_sub=gl.id_sub join groups g
on g.id_gr=gl.id_gr
join KindLearn k on k.id_kind=gl.id_kind
```

```
left outer join IsSubGr ifl on gl.id_sub=ifl.id_sub and
gl.id_gr=ifl.id_gr and gl.autumn=ifl.autumn and
     gl.learnYear=ifl.learnYear
left outer join GroupLoadTeachView v on gl.id_gl=v.id_gl
where gl.id_kind in(2,3) and yes is null
union
select  gl.id_sub,gl.autumn,gl.learnYear,short_name_sub,name_kind,
name_g=name_gr+'а',Prep=case when id_teacher is null then '' else
Teach end,id_teacher, gl.id_gr,s.id_dep
from Subject s join GroupLoad gl on s.id_sub=gl.id_sub join groups g
on g.id_gr=gl.id_gr
join KindLearn k on k.id_kind=gl.id_kind
 join IsSubGr ifl on gl.id_sub=ifl.id_sub and gl.id_gr=ifl.id_gr and
gl.autumn=ifl.autumn and gl.learnYear=ifl.learnYear
left outer join GroupLoadTeachView v on gl.id_gl=v.id_gl
where gl.id_kind=3 and v.subgr=0
union
select
gl.id_sub,gl.autumn,gl.learnYear,short_name_sub,name_kind,name_g=name_
gr+'б',Prep=case when id_teacher is null then '' else Teach end,
id_teacher, gl.id_gr,s.id_dep
from Subject s join GroupLoad gl on s.id_sub=gl.id_sub join groups g
on g.id_gr=gl.id_gr join KindLearn k on k.id_kind=gl.id_kind
 join IsSubGr ifl on gl.id_sub=ifl.id_sub and gl.id_gr=ifl.id_gr and
gl.autumn=ifl.autumn and   gl.learnYear=ifl.learnYear
left outer join GroupLoadTeachView v on gl.id_gl=v.id_gl
where gl.id_kind=3 and v.subgr=1

CREATE view ViewForDepSubLoadHourTemp
as
select l.autumn,gl.id_sub,gl.id_gr,id_fl,kol,HourLect=hours,
HourPract=0,HourLab=0,Exam=0,Zalik=0, Modul=0,Curs=0,RGR=0,
LearnPract=0,Consult=0,
id1=case when id_fl is null then gl.id_gr else id_fl end,
IsFl = case when id_fl is null then 0 else 1 end,
nameFl=case when id_fl is null then name_gr  else name_fl end
,s.id_dep, gl.LearnYear
from GroupLoad gl join Subject s on s.id_sub=gl.id_sub join groups g
on g.id_gr=gl.id_gr
left outer join IsFlowView ifl on gl.id_gr=ifl.id_gr and
gl.id_sub=ifl.id_sub  and gl.LearnYear=ifl.LearnYear and
gl.autumn=ifl.autumn  where gl.id_kind=1
union
select gl.autumn,gl.id_sub,gl.id_gr,id_fl,kol=0,
 HourLect=0,HourPract=hours,HourLab=0,Exam=0,Zalik=0,
     Modul=0,Curs=0,RGR=0,LearnPract=0,Consult=0,
     id1=case when id_fl is null then gl.id_gr else id_fl end,
     IsFl = case when id_fl is null then 0 else 1 end,
```

```
     nameFl=case when id_fl is null then name_gr  else name_fl end
   ,s.id_dep, gl.LearnYear
from GroupLoad gl join     Subject s on s.id_sub=gl.id_sub join groups
g on g.id_gr=gl.id_gr left outer join IsFlowView ifl on
gl.id_gr=ifl.id_gr and gl.id_sub=ifl.id_sub
and gl.LearnYear=ifl.LearnYear and gl.autumn=ifl.autumn
where gl.id_kind=2
union
select
gl.autumn,gl.id_sub,gl.id_gr,id_fl,kol=0,HourLect=0,HourPract=0,
     HourLab=hours*(1+ case when Sub_Gr='нет' then 0 else 1 end),
     Exam=0,Zalik=0,Modul=0,Curs=0,RGR=0,LearnPract=0,Consult=0,
     id1=case when id_fl is null then gl.id_gr else id_fl end,
     IsFl = case when id_fl is null then 0 else 1 end,
     nameFl=case when id_fl is null then name_gr  else name_fl end
   ,s.id_dep, gl.LearnYear
from GroupLoad gl join Subject s on s.id_sub=gl.id_sub join groups g
on g.id_gr=gl.id_gr
left outer join IsSubGrView isg on gl.id_gr=isg.id_gr and
gl.id_sub=isg.id_sub  and gl.LearnYear=isg.LearnYear and
gl.autumn=isg.autumn
     left outer join IsFlowView ifl on gl.id_gr=ifl.id_gr and
gl.id_sub=ifl.id_sub  and gl.LearnYear=ifl.LearnYear and
gl.autumn=ifl.autumn  where gl.id_kind=3
union
select
gl.autumn,gl.id_sub,gl.id_gr,id_fl,kol=0,HourLect=0,HourPract=0,HourLa
b=0,Exam=0,Zalik=0,
     Modul=0,Curs=0,RGR=0,LearnPract=0,Consult=hours,
     id1=case when id_fl is null then gl.id_gr else id_fl end,
     IsFl = case when id_fl is null then 0 else 1 end,
     nameFl=case when id_fl is null then name_gr  else name_fl end
   ,s.id_dep, gl.LearnYear
from GroupLoad gl join     Subject s on s.id_sub=gl.id_sub join groups
g on g.id_gr=gl.id_gr
left outer join IsFlowView ifl on gl.id_gr=ifl.id_gr and
gl.id_sub=ifl.id_sub  and gl.LearnYear=ifl.LearnYear and
gl.autumn=ifl.autumn  where gl.id_kind=4
union
select
gl.autumn,gl.id_sub,gl.id_gr,id_fl,kol=0,HourLect=0,HourPract=0,HourLa
b=0,Exam=hours,Zalik=0,
     Modul=0,Curs=0,RGR=0,LearnPract=0,Consult=0,
     id1=case when id_fl is null then gl.id_gr else id_fl end,
     IsFl = case when id_fl is null then 0 else 1 end,
     nameFl=case when id_fl is null then name_gr  else name_fl end
   ,s.id_dep, gl.LearnYear
```

```
from GroupLoad gl join Subject s on s.id_sub=gl.id_sub join groups g
on g.id_gr=gl.id_gr
     left outer join IsFlowView ifl on gl.id_gr=ifl.id_gr and
gl.id_sub=ifl.id_sub  and gl.LearnYear=ifl.LearnYear and
gl.autumn=ifl.autumn  where gl.id_kind=5
union
select
gl.autumn,gl.id_sub,gl.id_gr,id_fl,kol=0,HourLect=0,HourPract=0,HourLa
b=0,Exam=0,Zalik=hours,
     Modul=0,Curs=0,RGR=0,LearnPract=0,Consult=0,
     id1=case when id_fl is null then gl.id_gr else id_fl end,
     IsFl = case when id_fl is null then 0 else 1 end,
     nameFl=case when id_fl is null then name_gr  else name_fl end
   ,s.id_dep, gl.LearnYear
from GroupLoad gl join     Subject s on s.id_sub=gl.id_sub join groups
g on g.id_gr=gl.id_gr
left outer join IsFlowView ifl on gl.id_gr=ifl.id_gr and
gl.id_sub=ifl.id_sub  and gl.LearnYear=ifl.LearnYear and
gl.autumn=ifl.autumn  where gl.id_kind=6
union
select
gl.autumn,gl.id_sub,gl.id_gr,id_fl,kol=0,HourLect=0,HourPract=0,HourLa
b=0,Exam=0,Zalik=0,
     Modul=0,Curs=hours,RGR=0,LearnPract=0,Consult=0,
     id1=case when id_fl is null then gl.id_gr else id_fl end,
     IsFl = case when id_fl is null then 0 else 1 end,
     nameFl=case when id_fl is null then name_gr  else name_fl end
   ,s.id_dep, gl.LearnYear
from GroupLoad gl join     Subject s on s.id_sub=gl.id_sub join groups
g on g.id_gr=gl.id_gr
left outer join IsFlowView ifl on gl.id_gr=ifl.id_gr and
gl.id_sub=ifl.id_sub  and gl.LearnYear=ifl.LearnYear and
gl.autumn=ifl.autumn  where gl.id_kind=7
union
select
gl.autumn,gl.id_sub,gl.id_gr,id_fl,kol=0,HourLect=0,HourPract=0,HourLa
b=0,Exam=0,Zalik=0,
     Modul=0,Curs=0,RGR=0,LearnPract=hours,Consult=0,
     id1=case when id_fl is null then gl.id_gr else id_fl end,
     IsFl = case when id_fl is null then 0 else 1 end,
     nameFl=case when id_fl is null then name_gr  else name_fl end
   ,s.id_dep, gl.LearnYear
from GroupLoad gl join Subject s on s.id_sub=gl.id_sub join groups g
on g.id_gr=gl.id_gr
     left outer join IsFlowView ifl on gl.id_gr=ifl.id_gr and
gl.id_sub=ifl.id_sub  and gl.LearnYear=ifl.LearnYear and
gl.autumn=ifl.autumn  where gl.id_kind=8
union
```

```
select
gl.autumn,gl.id_sub,gl.id_gr,id_fl,kol=0,HourLect=0,HourPract=0,HourLa
b=0,Exam=0,Zalik=0,
      Modul=0,Curs=0,RGR=hours,LearnPract=0,Consult=0,
      id1=case when id_fl is null then gl.id_gr else id_fl end,
      IsFl = case when id_fl is null then 0 else 1 end,
      nameFl=case when id_fl is null then name_gr  else name_fl end
   ,s.id_dep, gl.LearnYear
from GroupLoad gl join     Subject s on s.id_sub=gl.id_sub join groups
g on g.id_gr=gl.id_gr
      left outer join IsFlowView ifl on gl.id_gr=ifl.id_gr and
gl.id_sub=ifl.id_sub  and gl.LearnYear=ifl.LearnYear and
gl.autumn=ifl.autumn  where gl.id_kind=9
union
select
gl.autumn,gl.id_sub,gl.id_gr,id_fl,kol=0,HourLect=0,HourPract=0,HourLa
b=0,Exam=0,Zalik=0,
      Modul=hours,Curs=0,RGR=0,LearnPract=0,Consult=0,
      id1=case when id_fl is null then gl.id_gr else id_fl end,
      IsFl = case when id_fl is null then 0 else 1 end,
      nameFl=case when id_fl is null then name_gr  else name_fl end
   ,s.id_dep, gl.LearnYear
from GroupLoad gl join     Subject s on s.id_sub=gl.id_sub join groups
g on g.id_gr=gl.id_gr
      left outer join IsFlowView ifl on gl.id_gr=ifl.id_gr and
gl.id_sub=ifl.id_sub  and gl.LearnYear=ifl.LearnYear and
gl.autumn=ifl.autumn  where gl.id_kind=10

CREATE view ViewAllSubLoadDep as
select autumn,short_name_sub,name_fl=nameFl,kol=sum(kol),
HLect=max(HourLect),HPract=sum(HourPract), HLab=sum(HourLab), HEx-
am=sum(Exam),HZalik=sum(Zalik),HModul=sum(Modul), HCurs=sum(Curs),
H_RGR=sum(RGR),H_LP=sum(LearnPract),HCons=sum(Consult),
V.id_sub,id1,IsFl, s.id_dep, V.LearnYear
from ViewForDepSubLoadHourTemp V join Subject s on V.id_sub=s.id_sub
group by autumn,short_name_sub,nameFl,V.id_sub,id1,IsFl, s.id_dep,
V.LearnYear
```

ДОДАТОК Г – ЗБЕРЕЖЕНІ ПРОЦЕДУРИ

```
CREATE proc FillGroupLoadDep
@id_dep tinyint,
@LearnYear smallint,
@k tinyint output,
@msg varchar (100) output
as
begin
      declare @@id_sub smallint, @@msg varchar (100), @@id_kind tinyint
      declare @@id_plan smallint,@@n smallint, @@k smallint
      set @k=0
      set @msg='OK'
      set @@n=0
      SET @@id_kind=(select id_kind from KindLearn where
name_kind='Модули')

      DECLARE SubLoad_cursor CURSOR FAST_FORWARD FOR
      SELECT s.id_sub,lp.id_plan
      From Subject s join LearnPlanSubject lps on s.id_sub=lps.id_sub
join
            LearnPlan lp on lps.id_plan=lps.id_plan
      Where s.id_dep=@id_dep and lp.LearnYear=@LearnYear
      OPEN SubLoad_cursor
      FETCH NEXT FROM SubLoad_cursor INTO @@id_sub,@@id_plan
      WHILE @@FETCH_STATUS = 0
      BEGIN
            delete GroupLoad
            where id_sub=@@id_sub and LearnYear=@LearnYear and
id_kind=@@id_kind

            insert into GroupLoad
            select id_gr, lps.id_sub, id_kind=@@id_kind, hours=modul,
            lps.id_plan,lps.autumn,LearnYear=@LearnYear,
            id_gl=convert(int,(@LearnYear-
2000)*power(2,25)+lps.id_sub*power(2,14)+id_gr*32+2*@@id_kind+
                        case when lps.autumn=1 then 1 else 0 end)
            from LearnPlanSubject lps join ViewSumModulSubGroup v
                  on lps.id_plan=v.id_plan and lps.autumn=v.autumn
                  and lps.id_sub=v.id_sub
            where lps.id_plan=@@id_plan AND lps.id_sub=@@id_sub
            set @@n=@@n+1
            FETCH NEXT FROM SubLoad_cursor INTO @@id_sub,@@id_plan
      END
      CLOSE SubLoad_cursor
      DEALLOCATE SubLoad_cursor

      set @@k=(select k=count(*) from GroupLoad
```

```
      where id_sub=@@id_sub and LearnYear=@LearnYear and
id_kind=@@id_kind)
      if @@k<@@n
      begin
            set @k=1
            set @msg='Не по всем дисциплинам кафедры назначены модули
CPC на '+  str(@LearnYear,4)+'-'+str(@LearnYear+1,4)+' уч.г.'
      end
end
GO


create   proc FillGroupLoadModul
@id_gr smallint,
@LearnYear smallint,
@k tinyint output,
@msg varchar (100) output
as
begin
      declare @@id_plan smallint, @@name_gr varchar(10), @@kol tinyint,
@@id_kind tinyint
--    declare @@Hour numeric(4,2), @@Hour2 numeric(4,2), @@percent
numeric (4,2)
      set @@name_gr=(select name_gr from groups where id_gr=@id_gr)

      if @@name_gr is null
      begin
            set @k=1
            set @msg = 'Группы с заданным идентификатором нет в БД'
      end
      else
      begin
          set @@id_plan= (select id_plan from GroupPlan
                      where id_gr=@id_gr and LearnYear=@LearnYear)
          if @@id_plan is null
          begin
            set @k=2
            set @msg = 'Группе '+@@name_gr+' не назначен учебный план
на'+STR(@LearnYear,4)+'-'+STR(@LearnYear+1,4)+' учебный год'
          end
          else
          begin
            set @@kol=(select kol from groups where id_gr=@id_gr)
            if @@kol>0
            begin
                  set @k=0
                  SET @@id_kind=(select id_kind from KindLearn where
name_kind='Модули')
```

```
                delete GroupLoad where id_gr=@id_gr and
LearnYear=@LearnYear and id_kind=@@id_kind
--нагрузка по модулям

                insert into GroupLoad
                select id_gr=@id_gr, lps.id_sub, id_kind=@@id_kind,
hours=modul,
                lps.id_plan,lps.autumn,LearnYear=@LearnYear,
                id_gl=convert(int,(@LearnYear-
2000)*power(2,25)+lps.id_sub*power(2,14)+@id_gr*32+2*@@id_kind+
                      case when lps.autumn=1 then 1 else 0 end)
                from LearnPlanSubject lps join ViewSumModulSubGroup v
                      on lps.id_plan=v.id_plan and lps.autumn=v.autumn
                      and lps.id_sub=v.id_sub
                where lps.id_plan=@@id_plan and id_gr=@id_gr
           end
           else
              begin
                set @k=3
                set @msg = 'У группы '+@@name_gr+' нулевой котингент
на '+STR(@LearnYear,4)+'-'+STR(@LearnYear+1,4)+' учебный год'
              end
         end

     end
end
GO

create proc FillDepSubLoadHourTemp
  @id_dep tinyint,
  @LearnYear smallint
as
begin
     delete DepSubLoadHourTemp where id_dep=@id_dep

     insert into DepSubLoadHourTemp
     select
short_name_sub,name_fl,kol,HourLect=Hlect,HourPract=HPract,HourLab=HLa
b,Exam=HExam,

     Zalik=HZalik,Modul=HModul,Curs=HCurs,RGR=H_RGR,LearnPract=H_LP,
          AllLoad= Hlect+ HPract+HLab +HExam +HZalik +HModul +
H_RGR+H_LP + HCons ,
          id_sub,id_fl=null,id_gr=null,id_dep,autumn,
          Consult=HCons
     from ViewAllSubLoadDep
     where id_dep=@id_dep and LearnYear=@LearnYear
```

```
      order by autumn desc
end
GO


CREATE proc FillTeachSubLoadHourTemp
  @id_dep tinyint,
  @LearnYear smallint
as
begin
      delete TeachSubLoadHourTemp where id_dep=@id_dep

      insert into TeachSubLoadHourTemp
      select
v.short_name_sub,name_fl,kol=0,HourLect=Hlect,HourPract=HPract,HourLab
=HLab,Exam=HExam,

      Zalik=HZalik,Modul=HModul,Curs=HCurs,RGR=H_RGR,LearnPract=H_LP,
            AllLoad= Hlect+ HPract+HLab +HExam +HZalik +HModul +
H_RGR+H_LP + HCons ,
            v.id_sub,id_fl=null,id_gr=null,id_dep,id_teach, autumn,
            Consult=HCons
      from ViewAllSubLoadTeach v join Subject s on v.id_sub=s.id_sub
      where id_dep=@id_dep and LearnYear=@LearnYear
      order by autumn desc
end
GO
```

ДОДАТОК Д ВИХІДНИЙ КОД ЗАСТОСУВАННЯ ПРАЦІВНИКА КАФЕДРИ