МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської та аспірантської підготовки

Кафедра інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: МОДЕЛЮВАННЯ ТА РОЗРОБКА ПІДСИСТЕМИ «МЕТОДИЧНИЙ ВІДДІЛ» У СКЛАДІ ІНФОРМАЦІЙНОЇ СИСТЕМИ «НАВЧАЛЬНИЙ ПРОЦЕС УНІВЕРСИТЕТУ»

Виконав студент 2 року групи МК- 1 спеціальності 122 Комп’ютерні науки

Котошенко Віталій В’ячеславович

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

Консультант

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

Одеса 2018

АНОТАЦІЯ

Магістерська робота на тему «Моделювання та розробка підсистеми «Методичний відділ» інформаційної системи «Навчальний процес університету»» обумовлена задачею автоматизації основних робіт, які забезпечують навчальний процес у вищому навчальному закладі. Це визначає актуальність даної роботи.

Наукова новизна роботи полягає в тому, що у даний роботі розглядається весь навчальний процес цілком, з урахуванням взаємодії та зв'язків окремих складових частин цього процесу. Дана магістерська робота розглядає важливу складову частину навчального процесу – забезпечення методичної підтримки навчання у закладі вищої освіти. Цю роботу виконують працівники методичного відділу закладу вищої освіти.

Метою і задачею дослідження є вивчення методів та моделей автоматизації робіт, що забезпечують методичну підтримку навчального процесу у закладі вищої освіти.

Об'єктом дослідження є навчальний процес у закладі вищої освіти та його методичне забезпечення.

Предметом дослідження є засоби автоматизації даного процесу.

Вихідні дані**.** Використовуються дані, необхідні для проведення навчального процесу у вищі: списки факультетів закладу вищої освіти, списки кафедр, списки навчальних планів, навчальних дисциплін, тощо.

Результати даної роботи можуть використовуватись для автоматизації роботи працівника методичного відділу при забезпеченні методичної підтримки навчального процесу у будь-якому закладі вищої освіти, що використовує кредитно-модульну систему навчання.

Ключові слова: ІНФОРМАЦІЙНА СИСТЕМА, КОНТЕКСТНА ДІАГРАМА IDEF0, ER-ДІАГРАМА, РЕЛЯЦІЙНА БАЗА ДАНИХ, СЕРВЕРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, НАВЧАЛЬНО-МЕТОДИЧНИЙ КОМПЛЕКС, АВТОМАТИЗОВАНЕ РОБОЧЕ МІСЦЕ.

Обсяг роботи 60 сторінок. Вона містить 27 рисунків та 15 посилань.

SUMMARY

The master's work on the topic "Modeling and development of the subsystem "Methodical Department" as part of the information system "Educational process of the university"" is due to the task of automating the basic work that provides the educational process at the university. This determines the relevance of this work.

The scientific novelty of the work lies in the fact that in the present work the entire educational process is considered in its entirety, taking into account the interactions and connections of the individual components of this process. This master's work considers an important part of the educational process - providing methodological support for learning in a higher education institution. This work is performed by employees of the methodological department of the institution of higher education.

The purpose and objective of the study is to study the methods and models of work automation that provide methodological support for the educational process in a higher education institution.

The object of research is the educational process in the institution of higher education and its methodological support.

The subject of research is the automation of this process.

Initial data. The data required for the educational process at the university are used: lists of faculties of institutions of higher education, lists of departments, lists of teachers of institutions of higher education, academic groups, curricula, academic disciplines and the like.

The results of this work can be used to automate the work of an employee of the methodical department while providing methodological support for the educational process in any institution of higher education where a credit-module system of education is used.

Keywords: INFORMATION SYSTEM, IDEF0 CONTEXT DIAGRAM, ER-DIAGRAM, RELATIONAL DATABASE, SERVER SOFTWARE, EDUCATIONAL-METHODICAL COMPLEX, AUTOMATED WORKPLACE.

The amount of work 60 pages. It contains 27 figures and 15 references.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....7
ВСТУП 8
1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ 10
1.1 Об’єкти та суб’єкти навчального процесу університету 10
2 МЕТОДОЛОГІЯ ФУНКЦІОНАЛЬНОГО МОДЕЛЮВАННЯ (IDEF0) 12
3 СИСТЕМНИЙ АНАЛІЗ І КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....15
3.1 Етапи проектування бази даних 15
3.2 Концептуальне проектування бази даних 16
3.2.1 Визначення основних типів сутностей та типів зв’язків 16
3.2.2 Визначення атрибутів і зв’язування їх з типами сутностей і зв’язків 27
3.2.3 Визначення атрибутів, що є потенційними і первинними ключами28
3.2.4 Визначення відповідності концептуальної моделі транзакціям користувачів.....28
4 ВИБІР СУБД ТА СЕРЕДОВИЩА РОЗРОБКИ ЗАСТОСУВАННЯ.....32
4.1 Вибір СУБД.....32
4.2 Вибір середовища розробки застосування та мови програмування 32
4.2.1 Мова програмування C# 34
5 ЛОГІЧНЕ І ФІЗИЧНЕ ПРОЕКТУВАННЯ БД 36
5.1 Логічне проектування.....36
5.2 Фізичне проектування 36
6 СЕРВЕРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.....38
7 ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ 44
7.1 Загальні відомості 44
7.2 Функціональне призначення 44
7.3 Керівництво програміста 44
7.4 Посібник користувача 48
ВИСНОВКИ.....58
ПЕРЕЛІК ПОСИЛАНЬ.....60
Д О Д А Т К И ДОДАТОК А Логічна схема бази даних 61
ДОДАТОК А Логічна схема бази даних.....62
ДОДАТОК Б Фізична схема бази даних.....63
ДОДАТОК В Серверне програмне забезпечення 64
ДОДАТОК Г Вихідний код клієнтського застосування 67

ПЕРЕЛІК СКОРОЧЕНЬ

АРМ – автоматизоване робоче місце
БД – база даних
ЗВО – заклад вищої освіти
ІС – інформаційна система
МОН – міністерство освіти та науки
НМК – навчально-методичний комплекс
ОПП – освітньо-професійна програма підготовки фахівців
ОНП – освітньо-наукова програма підготовки фахівців
СРС – самостійна робота студентів
СУБД – система управління базами даних

ВСТУП

Методичний відділ університету повинен забезпечувати методичну підтримку у виконанні закладом вищої освіти основної задачі – наданні студентам можливості отримання вищого рівня освіти. Працівники методичного відділу розробляють встановлюють перелік методичних документів, що входять до навчально-методичного комплексу (НМК) забезпечення навчальної дисципліни. Також працівники цього відділу забезпечують вчасне періодичне оновлення всіх складових НМК, перевіряють та затверджують всі методичні документи вищого навчального закладу: освітньо-професійні програми (ОПП) підготовки фахівців та освітньо-наукові програми (ОНП); профілі спеціальностей, за якими навчаються студенти ЗВО; навчальні плани всіх спеціальностей всіх факультетів, робочі програми навчальних дисциплін, тощо.

Розклад занять на наступний семестр складається згідно робочим навчальним планам, за якими навчаються академічні групи університету. Для можливості складання розкладу занять навчальний відділ отримує від кафедр університету навчальне навантаження викладачів, тобто розподіл аудиторних занять між викладачами кафедри. Деканати факультетів надають до навчального відділу контингент студентів – кількісний склад всіх академічних груп, що будуть навчатись у наступному семестрі.

На протязі навчального року викладачі проводять аудиторні заняття в академічних групах, консультації, забезпечують поточний контроль знань студентів. Виконанню обов'язків викладачів сприяє методична підтримка у вигляді навчально-методичного комплексу дисципліни.

З переліку вихідних даних, необхідних для автоматизованого роботи методичного відділу зрозуміло, що недоцільно розробляти окрему інформаційну систему (ІС), спрямовану лише для вирішення цієї приватної задачі, оскільки робота цього підрозділу університету зв’язана з діяльністю інших підрозділів, з якими відбувається постійний обмін документами.

Потрібно розглядати задачу розробки автоматизованого робочого місця (АРМ) працівника методичного відділу у контексті єдиної інформаційної системи «Навчальний процес університету». Інформаційна система, яка працює з усіма складовими навчального процесу університету, дозволяє користувачам оперувати тільки тими даними, що відносяться до їх компетенції, а загальні дані, наприклад, основні таблиці-довідники з переліком підрозділів університету, вносить у базу даних адміністратор ІС.

Таким чином, при проектуванні вказаної єдиної інформаційної системи виявляється, що вона повинна містити багато складових частин, також виявляється декілька груп користувачів, які будуть працювати з цією системою.

Задачею магістерської роботи є моделювання та проектування підсистеми «Методичний відділ» у складі ІС «Навчальний процес університету», та розробка застосування «АРМ працівника методичного відділу» у якості додатку до підсистеми «Навчальний відділ».

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Головною задачею вищого навчального закладу є надання студентам можливості отримання вищої освіти рівня бакалавр або магістр. Навчальний процес є основним у вищі, і всі інші види діяльності не можливі без існування цього процесу. При створенні інформаційної системи університету у першу чергу потрібно включити до неї саме навчальний процес.

При проектуванні ІС «Навчальний процес університету» необхідно розглянути всі підсистеми, які беруть участь у даному процесі. Також потрібно виявити всіх можливих учасників цього процесу, визначити всі групи користувачів.

1.1 Об’єкти та суб’єкти навчального процесу університету

Метою існування вищих навчальних закладів є надання студентам можливості отримання вищого рівня освіти. Таким чином, основним суб’єктом навчального процесу є студент, який має бажання отримати цю освіту. Можливість здобуття освіти надається студенту за допомогою спілкування з викладачем, який є другим основним суб'єктом навчального процесу.

На протязі навчального року викладачі проводять аудиторні заняття в академічних групах, консультації, забезпечують поточний контроль знань студентів. Виконанню обов'язків викладачів сприяє методична підтримка у вигляді навчально-методичного комплексу дисципліни.

Освіту студент набуває за допомогою аудиторних занять та самостійної роботи. Для можливості ефективної самостійної підготовки студентів викладачі розробляють методичне забезпечення, склад якого визначає методичний відділ.

Студенті навчаються за вибраною спеціальністю у відповідності до профілю спеціальності та навчальному плану навчання фахівців з цієї спеціальності. Профілі спеціальностей розробляються проектними групами та затверджуються методичним відділом, так саке, як освітньо-професійні програм (ОПП) та освітньо-наукові програми (ОНП) навчання зі спеціальності.

Робочі навчальні плани розроблять деканати факультетів на основі ОПП та ОНП. Розроблені навчальні плани також розглядаються та затверджуються методичним відділом університету.

Навчальні плани містять перелік дисциплін, які повинні вивчати студенти, що навчаються за даним навчальним планом. Методичний відділ університету складає перелік всіх дисциплін, що будуть вивчатись у наступному навчальному році, та розподіляє ці дисципліни між кафедрами університету.

Інформаційна система «Навчальний процес університету» дозволить спростити потік документів між підрозділами університету.

Автоматизація процесу планування методичного навантаження викладачів та підрахунку виконаного методичного навантаження значно спростить функції працівників методичного відділу по контролю за планами методичного навантаженні викладачів та результатами їх виконання.

2 МЕТОДОЛОГІЯ ФУНКЦІОНАЛЬНОГО МОДЕЛЮВАННЯ (IDEF0)

IDEF0 – це діаграма, розташована на вершині деревовидної структури діаграм, що представляє собою саме загальний опис системи та її взаємодію з зовнішнім середовищем. Контекстна діаграма складається з одного блоку, що описує функцію верхнього рівня, її входи, виходи, управління, і механізми, разом з формулюваннями мети моделі і точки зору, з якої будується модель [1 – 2].

Головною функцією системи, що моделюється, є забезпечення навчального процесу в університеті.

Список даних: ОПП, ОНП, нормативні документи МОН та ЗВО, студент, викладач, навчальний план, аудиторний фонд, вимоги викладачів до розкладу занять, розклад занять, проведення занять.

Список функцій: розробка навчальних планів за спеціальностями, розподіл занять та іншого навчального навантаження між викладачами, складання розкладу занять, проведення аудиторних занять, проведення заходів заключного контролю – іспит або залік, підведення підсумків виконання навантаження викладачами.

Мета створення інформаційної системи: Організувати проведення навчального процесу в університеті.

Перелік запитань:

– За якими нормативними документами розробляються навчальні плани?
– Які навчальні плани повинні бути у наступному навчальному році?
– Які академічні групи навчаються за яким навчальним планом?
– Які документи необхідні для методичного забезпечення навчального процесу в університеті?
– Які методичні документи складає навчально-методичний комплекс навчальної дисципліни?
– Яке методичне навантаження заплановане викладачам кафедр на наступний навчальний рік?
– Як викладачі виконали методичне навантаження за минулий рік?

У контекстній діаграмі будуть присутніми такі граничні стрілки:

1) Входи: студент, викладач.
2) Управління: нормативні документи МОН та ЗВО, ООП, ОНП, аудиторний фонд.
3) Механізми: деканат, методичний відділ, кафедра, навчальний відділ.

4) Виходи: успішність студента.

Головною задачею закладу вищої освіти (ЗВО) є надання студентам можливості отримання вищої освіти рівня бакалавр або магістр. При створенні інформаційної системи університету у першу чергу потрібно включити до неї саме навчальний процес.

Студенти ЗВО здобуваються освіту з вибраної спеціальності згідно освітньо-професійній програмі підготовки фахівців (ОПП). На основі ОПП уповноваженими комісіями розробляються навчальні плани навчання за спеціальностями. Ці плани уточнюються деканатами факультетів і затверджуються методичним відділом університету.

Навчальні плани містять перелік дисциплін, які повинні вивчити студенти, загальний обсяг годин на вивчення кожної дисципліни, та розподіл годин між різними видами занять та самостійною підготовкою студентів. Навчальні дисципліни розподіляються між кафедрами університету. На кафедрах для кожної дисципліни призначається провідний викладач, якій відповідає за якість підготовки фахівців з цієї дисципліни. Провідним викладачем розробляється робоча програма з дисципліни на основі навчального плану. В робочій програмі з дисципліни вказуються тематика занять, кількість і вид контролюючих заходів. Також провідний викладач розробляє графік проведення контролюючих заходів протягом семестру. Методичний відділ перевіряє та затверджує робочі програми з дисциплін.

Для забезпечення навчального процесу викладачі повинні розробляти різні навчально-методичні матеріали з дисциплін: конспекти лекцій, навчальні пособники, методичні вказівки для виконання різних видів занять (практичних занять, лабораторних занять, курсових робіт та проектів, завдань для самостійної підготовки студентів).

При моделюванні інформаційної системи діаграма IDEF0 представляє собою загальний опис системи та її взаємодію з зовнішнім середовищем. Контекстна діаграма складається з одного блоку, що описує функцію верхнього рівня, її входи, виходи, управління, і механізми, разом з формулюваннями мети моделі і точки зору, з якої будується модель.

З опису навчального процесу ЗВО зрозуміло, що декомпозиція контекстної діаграми IDEF0 повинна включати блоки розробки навчальних планів, розподілу навчального навантаження між викладачами, складання розкладу занять, проведення поточного навчального процесу (рис. 2.1).

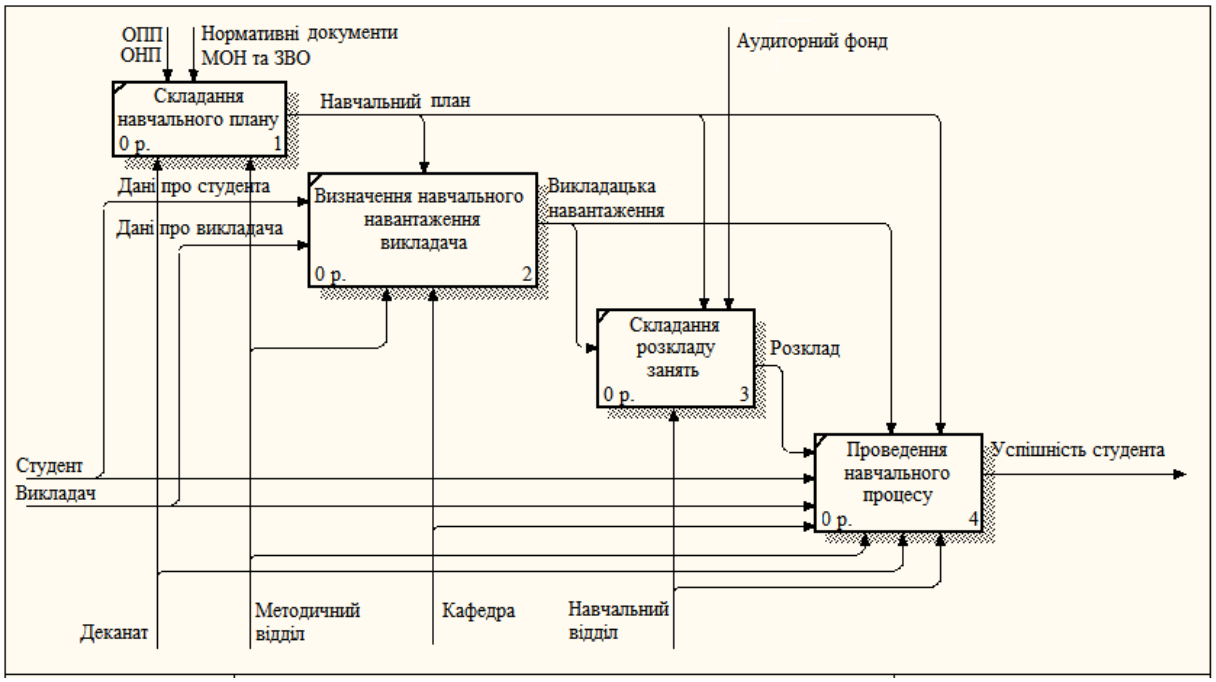

Рисунок 2.1 – Декомпозиція контекстної IDEF0 діаграми «Забезпечення навчального процесу»

Інформаційна система повинна ґрунтуватися на базі даних, яка буде працювати під управлінням СУБД, що забезпечить можливість працювати одночасно різним групам користувачів та дозволить надати цим групам користувачів права та привілеї, які необхідні їм для роботи.

Однією з груп користувачів буде група «Працівник методичного відділу». Для роботи користувачів цієї групи з ІС потрібно розробити програмне застосування, яке забезпечить можливість виконання користувачами всіх необхідних ним транзакцій.

3 СИСТЕМНИЙ АНАЛІЗ І КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Інформаційна система повинна ґрунтуватися на базі даних з кількох причин. По-перше, в навчальному процесі ЗВО задіяні великі обсяги даних. По-друге, ці дані повинні використовувати різні групи користувачів системи. По-третє, ці користувачі повинні мати можливість працювати з системою одночасно. Тому необхідно спроектувати базу даних інформаційної системи, яка буде працювати під управлінням СУБД, що забезпечить можливість працювати одночасно різним групам користувачів та дозволить надати цим групам користувачів права та привілеї, які необхідні їм для роботи.

Проектування бази даних починається зі збору зовнішніх уявлень на базу даних всіх груп користувачів.

3.1 Етапи проектування бази даних

Процес проектування бази даних складається з трьох основних етапів: концептуальне, логічне і фізичне проектування [3 – 4].

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

Основні етапи концептуального проектування [3]:

- інтеграція зовнішніх представлень користувачів;
- 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 транзакціям 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 типів зв'язків «багато до багатьох»;
- визначення набору відношень, виходячи зі структури локальної логічної моделі даних;
- перевірка відношень за допомогою правил нормалізації;
- перевірка відповідності відношень вимогам транзакцій всіх груп користувачів;
- 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 відношень у середовищі обраної СУБД;
- проектування похідних відношень;
- реалізація обмежень цілісності предметної області;;
- визначення індексів;
- розробка уявлень користувачів.

3.2 Концептуальне проектування бази даних

3.2.1 Визначення основних типів сутностей та типів зв’язків

З опису предметної області та з діаграмі декомпозиції IDEF0 можна виділити основні групи користувачів ІС «Навчальний процес університету» і їх транзакції.

Основними групами користувачів будуть:

- працівник деканату;
- працівник методичного відділу;
- працівник кафедри;
- працівник навчального відділу;
- викладач;
- студент.

Для моделювання та розробки підсистеми «Навчальний відділ» потрібно спроектувати схему «Методичний відділ» як підсхему бази даних «Навчальний процес». Тому потрібно розглянути докладно основні транзакції групи користувачів «Працівник методичного відділу».

Основні транзакції користувачів даної групи:

1) Відновити нормативні документи щодо навчального процесу на наступний навчальний рік.
2) Отримати від уповноважених комісій ОПП та ОНП за спеціальностями.
3) Затвердити ОПП та ОНП за спеціальностями.
4) Отримати від факультетів робочі навчальні плани на наступний навчальний рік.
5) Затвердити робочі навчальні плани факультетів на наступний навчальний рік.
6) Скласти список дисциплін, що включені в навчальні плани на наступний рік та розподілити їх між кафедрами університету.
7) Отримати від провідних викладачів робочі програми з дисциплін.
8) Затвердити робочі програми з дисциплін.
9) Відновити список можливих контролюючих заходів для оцінки успішності навчання студентів.
10) Відновити перелік документів, що входять до НМК дисципліни.
11) Відновити перспективні плани оновлення НМК по дисциплінах кафедри.
12) Отримати від кафедр план розробки навчально-методичного забезпечення дисциплін на наступний навчальний рік.
13) Відновити таблиці розрахунку методичного навантаження викладача.
14) Затвердити плани розробки навчально-методичного забезпечення дисциплін на наступний навчальний рік.

15) Перевірити виконання викладачами заходів розробки навчально-методичного забезпечення дисциплін у попередньому навчальному році

Таким чином, виявляються основні базові типи сутностей в базі даних:

- «Факультет»;
- «Кафедра»;
- «Навчальний план»;
- «Викладач»;
- «Дисципліна (навчальна)»;
- «Робоча програма (дисципліни)».

Для проектування бази даних потрібно виявити зв’язки між базовими типами сутностей.

Факультет має декілька академічних груп та декілька навчальних планів – між типами сутностей «Факультет» та «Навчальний план» існує тип зв’язку «один до багатьох.

У навчальних планах перелічені дисципліни, що вивчаються протягом року студентами даного курсу даної спеціальності.

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 «Дисципліна плану» (пункт навчального плану, дисципліна навчального плану), відповідно до якого і потрібно визначати заняття з деякої дисципліни для деякої групи.

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

У однієї дисципліни навчального плану зазвичай заплановано декілька видів навчального навантаження: лекції, практичні заняття (семінари), лабораторні роботи.

Усі три види занять зустрічаються для однієї дисципліни досить рідко, зазвичай планується проведення лекцій та одного з видів практичних занять:

семінари або лабораторні роботи. Інколи для дисципліни можуть бути призначені тільки лекції, або тільки практичні заняття.

Отже, потрібен додатковий похідний тип сутностей «Вид занять дисципліни», який буде містити дані про те, якого виду заняття передбачені для кожної дисципліни, та кількість годин аудиторних занять цього виду призначено на семестр. Між типами сутностей «Дисципліна плану» та «Вид занять дисципліни» існує тип зв'язку «один до багатьох», оскільки у загальному випадку для однієї дисципліни протягом семестру проводяться заняття декількох видів.

По виявлених типам сутностей та визначеним типам зв'язків між ними можна побудувати першу концептуальну модель БД підсхеми «Методична робота» (рис. 3.1).

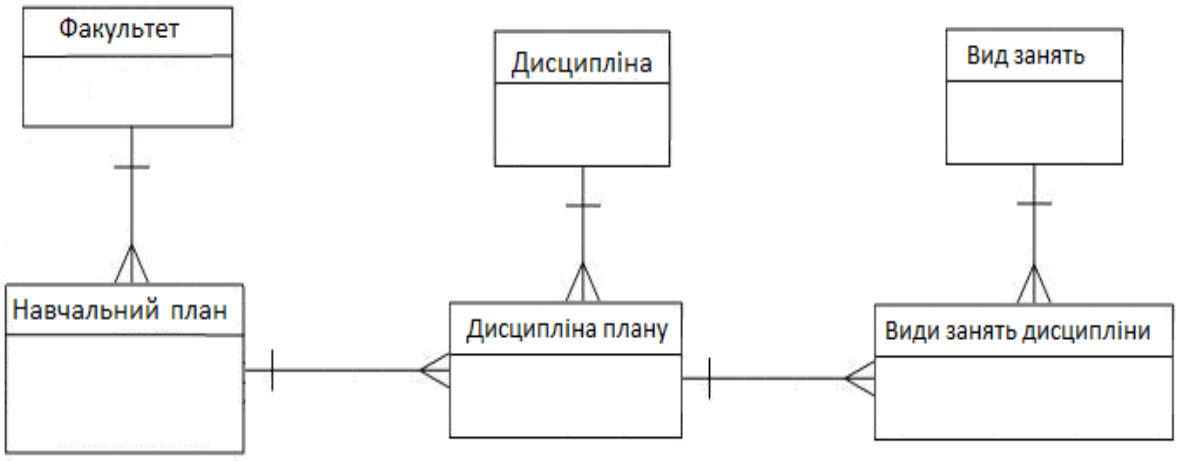

Рисунок 3.1 – Перша концептуальна модель БД підсхеми «Методична робота»

Дисципліни закріплюються за кафедрами. На кафедрі працює декілька викладачів. Кожен викладач веде одну або декілька дисциплін. Одну дисципліну можуть вести декілька викладачів.

На кожній кафедри для дисципліни призначається провідний викладач, який складає робочу програму дисципліни. Якщо дисципліна вивчається на декількох спеціальностях, для дисципліни може складатись кілька робочих програм і призначатись декілька провідних викладачів.

Таким чином для виявлених типів сутностей існують наступні типи зв'язків. Між типами сутностей «Кафедра» та «Викладач» існує тип зв'язку «один до багатьох». Між типами сутностей «Кафедра» і «Дисципліна» так само існує тип зв'язку «один до багатьох».

Оскільки для різних спеціальностей наповнення дисципліни може відрізнятись, доцільно зв’язувати викладача не з базовим типом сутностей «Дисципліна», а з асоціативним типом сутностей «Дисципліна плану».

Між типами сутностей «Викладач» та «Дисципліна плану» існує тип зв’язку «багато до багатьох»: викладач, як правило, веде декілька дисциплін; одну дисципліну одного навчального плану можуть вести декілька викладачів, наприклад, провідний викладача та асистент.

По доданим типам сутностей та визначеним типам зв’язків між ними можна побудувати доповнену концептуальну модель БД підсхеми «Методична робота» (рис. 3.2).

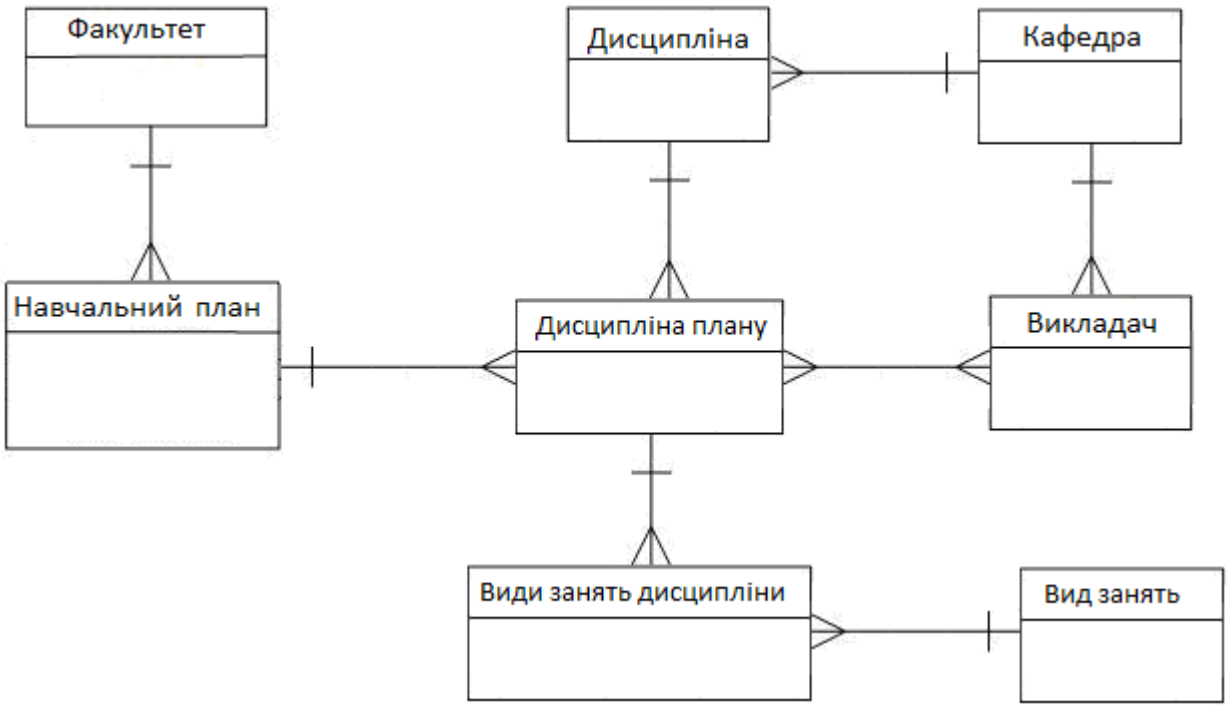

Рисунок 3.2 – Доповнена концептуальна модель БД підсхеми «Методична робота»

Існування типів зв’язків «багато до багатьох» ускладнює схему, робить неможливим використання реляційної моделі даних. Це обмеження можна усунути при побудові логічної моделі бази даних, якщо буде вибрана саме реляційна модель даних, але бажано вже на етапі концептуального проектування ввести асоціативний тип сутності «Викладач дисципліни». Цей асоціативний тип потрібен тому, що саме цеп тип сутності може мати атрибут, який вказує, чи є викладач провідним викладачем з дисципліни і

зобов’язаний методично забезпечувати дисципліни, зокрема складати для неї робочу програму.

Асоціативний тип сутності «Викладач дисципліни» повинен зв’язувати тип сутності «Викладач» з типом сутностей «Дисципліна плану». Між типом сутностей «Викладач» та типом сутностей «Викладач дисципліни» існує тип зв’язку «один до багатьох»; Між типами сутностей «Дисципліна плану» та «Дисципліна плану» також існує тип зв’язку «один до багатьох».

З типом сутностей «Викладач дисципліни» повинен бути зв’язаний тип сутностей «Робоча програма».

Визначивши типи зв'язків між виявленими типами сутностей, отримаємо уточнену концептуальну модель бази даних для підсхеми «Методична робота» (рис. 3.3).

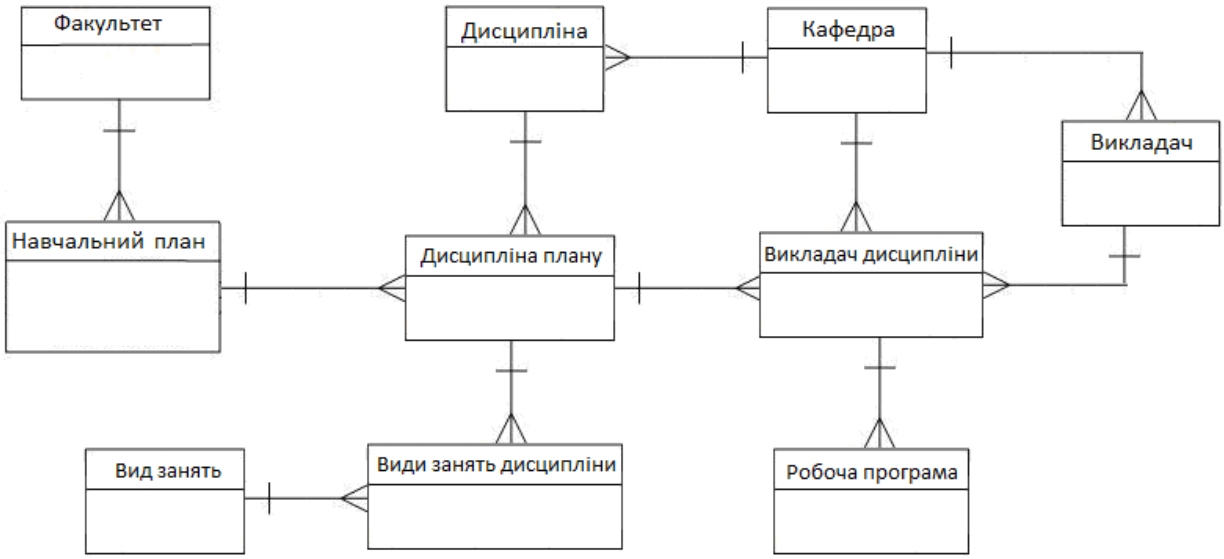

Рисунок 3.3 – Уточнена концептуальна модель БД
підсхеми «Методична робота»

В університеті навчання ведеться за кредитно-модульною системою. Кожній навчальній дисципліні виділяється задана кількість кредитів. Зараз кредит дорівнює 30 академічним годинам. З загальної кількості годин на вивчення дисципліни у робочому навчальному плані записується, яка кількість годин відводиться на аудиторні заняття різних видів. Решта годин відводиться на самостійну підготовку студентів.

Таким чином, тип сутності «Робоча програма» зв’язаний з типом сутностей «Вид занять дисципліни» типом зв’язків «один до багатьох».

У робочій програмі вивчення дисципліни розбивається на декілька модулів. Модулі можуть бути лекційними або практичними (рис. 3.4).

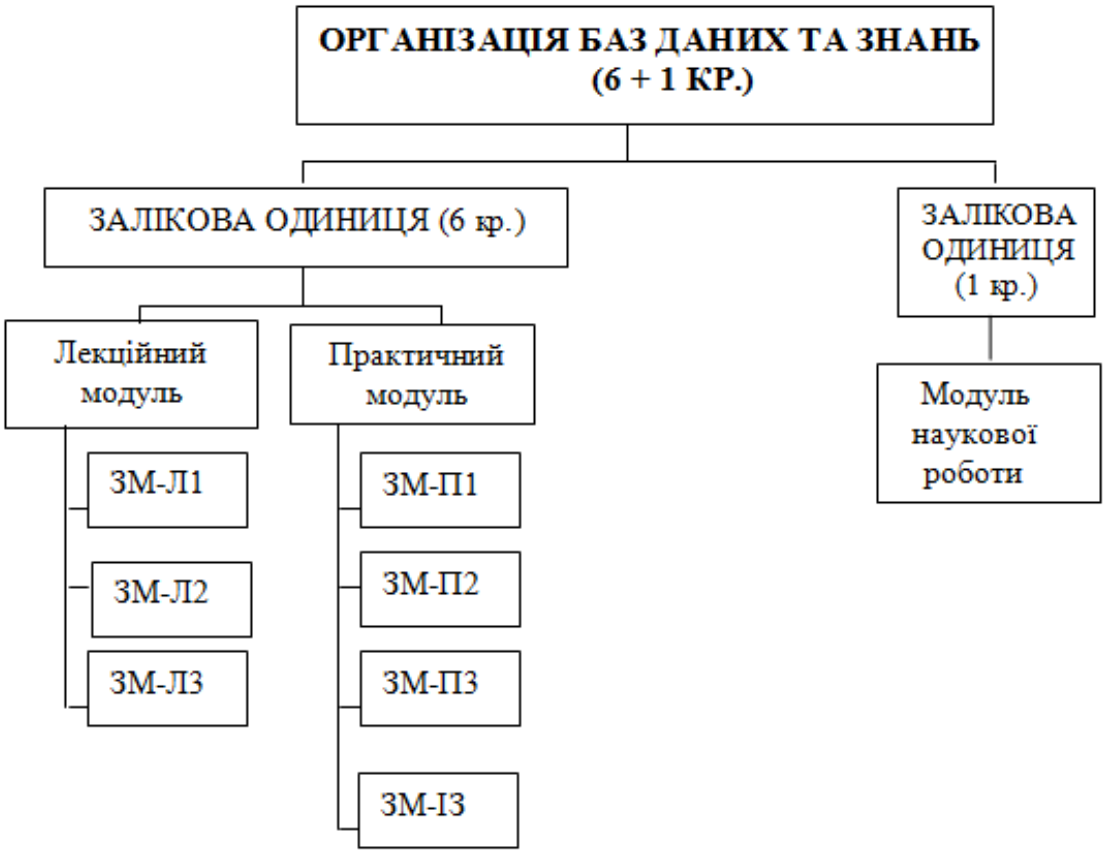

Рисунок 3.4 – Структура навчальної дисципліни у робочій програмі

Для лекційних модулів у робочій програмі записується перелік тем, визначається кількість академічних годин лекційних занять, та кількість годин, відведених на самостійну підготовку студентів (рис. 3.5).

Таким чином, виявляються нові типи сутностей для схеми «Методична робота»: «Модуль» та «Тема дисципліни». Тип сутностей «Модуль» зв’язаний з типом сутностей «Робоча програма»: як видно з рис. 3.4 – 3.5, між типами сутностей «Робоча програма» та «Модуль» існує тип зв’язку «один до багатьох».

Тип сутностей «Тема дисципліни» зв’язаний з типом сутностей «Модуль», і як видно з рис. 3.5, між типами сутностей «Модуль» та «Тема дисципліни» також існує тип зв’язку «один до багатьох».

На рис. 3.6 наведений опис практичних модулів в робочій програмі дисципліни. Як видно з цього рисунку, практичні модулі також мають декілька тем – теми робіт, які виконуються як складові практичного модулю.

Програма лекційних модулів

Змістовні модулі	Назва змістовного модуля	Назви тем	Денна форма			
			Кіл-сть аудиторних годин	Кіл-сть годин СРС	Форми завдань на СРС	Форми поточного контролю СРС
ЗМ-Л1	**Моделювання даних**	**Тема 1.** Системи баз даних. Основні поняття та архітектура	2	1	ПЛЗ ПМКР	КР-1
		Тема 2. Моделі даних	2	1	ПЛЗ ПМКР	
		Тема 3. Реляційна модель даних	4	2	ПЛЗ ПМКР	
		Тема 4. Методи аналізу та проектування баз даних	2	1+5	ПЛЗ ПМКР	
		Тема 5. Теорія нормалізації реляційної моделі	2	1	ПЛЗ ПМКР	
	Разом за модуль		12	11		
ЗМ-Л2	**Робота з реляційними базами даних**	**Тема 6.** Мова SQL	6	3	ПЛЗ ПМКР	КР-2
		Тема 7. Мова T-SQL	6	2	ПЛЗ ПМКР	
		Тема 8. Розробка застосувань до баз даних	8	2+5	ПЛЗ ПМКР	
	Разом за модуль		20	12		

Рисунок 3.5 – Опис лекційних модулів у робочій програмі дисципліни

ПРОГРАМА ПРАКТИЧНИХ МОДУЛІВ

Змістовні модулі	Форма занять (назва)	Теми робіт (занять)	Денна форма			
			Кіл-сть аудиторних годин	Кіл-сть годин СРС	Форми завдань на СРС	Форми поточного контролю СРС
ЗМ-П1	Лабораторні	1. Створення бази даних у реляційній СКБД MS SQL Server 2000.	2	2+1	ПУОП ПЛР	УО ЛР
		2. Прості запити до БД.	4	2+1		УО ЛР
		3. Складні запити до БД: вкладені запити та групування	4	2+1		УО ЛР
		4. Маніпулювання даними в БД командами мови Transact-SQL	4	2+1		УО ЛР
		Разом за модуль	**14**	**12**		
ЗМ-П1	Лабораторні	5. Проектування та розробка БД	4	2+1	ПУОП ПЛР	УО ЛР
		6. Розробка серверного ПЗ	6	2+1		УО ЛР
		7. Створення клієнтського додатку до БД	4	2+1		УО ЛР
		Разом за модуль	**14**	**9**		

Рисунок 3.6 – Опис практичних модулів у робочій програмі дисципліни

Також під кожну тему відводяться аудиторні години на виконання роботи та години самостійної роботи для підготовки до роботи та оформлення звіту про виконання роботи.

Лекційні або практичні модулі можуть бути присутні у робочій програмі дисципліни, якщо у навчальному плані передбачений для дисципліни даний вид занять.

Оскільки для одного виду занять у робочій програмі планується декілька модулів, а кожен модуль може відноситись лише до одного виду занять, то між типами сутностей «Вид заняття дисципліни» та «Модуль» існує тип зв’язку «Один до багатьох».

Для кожного модулю повинні передбачатись заходи поточного контролю самостійної роботи студентів (СРС). Як видно з рис. 3.4 – 3.5 захід поточного контролю може бути прив’язаний або до модулю, або до теми модулю. Тому потрібно ввести два похідних типа сутності «Контроль теми» та «Контроль модулю».

Тип сутності «Модуль» зв’язаний з типом сутностей «Контроль модулю» типом зв’язків «один до багатьох», оскільки для одного модулю може бути призначено декілька заходів поточного контролю, наприклад, контрольна робота та усне опитування.

Це стосується і типу зв’язку між типами сутностей «Тема дисципліни» та «Контроль теми».

Визначивши типи зв'язків між новими виявленими типами сутностей, отримаємо другу уточнену концептуальну модель бази даних для підсхеми «Методична робота» (рис. 3.7).

На підготовку до кожного заходу поточного контролю студентам потрібно відводити години СРС. Різні заходи контролю потребують різної кількості годин підготовки. Тому методичний відділ складає таблицю можливих заходів СРС та кількості годин, відведених на підготовку студентів до цих заходів. Тому у схему потрібно додати новий базовий тип сутності «Вид СРС».

Тип сутності «Вид СРС» (вид заходу контролю СРС) зв’язаний з похідним типом сутностей «Контроль модулю» типом зв’язку «один до багатьох»; також між типом сутності «Вид СРС» та похідним типом сутностей «Контроль теми» існує тип зв’язку «один до багатьох».

На проведення та перевірку заходів поточного контролю СРС викладачам відводяться години навантаження. Тому потрібен похідний тип сутності «Контроль викладача».

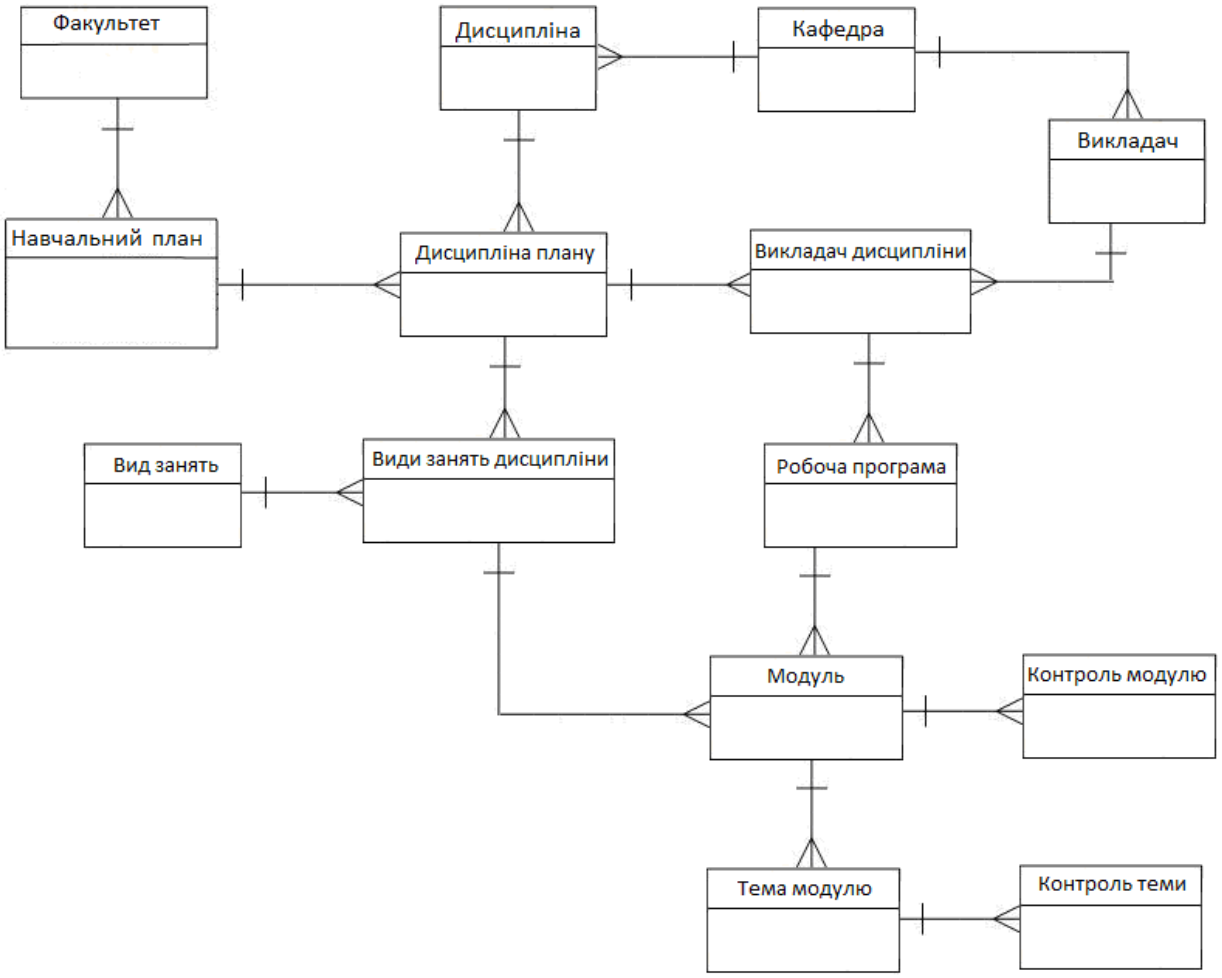

Рисунок 3.7 – Друга уточнена концептуальна модель бази даних для підсхеми «Методична робота»

Між типами сутностей «Викладач» та «Контроль викладача» існує тип зв’язку «один до багатьох». Якщо вважати, що кожен заклад поточного контролю поточного контролю проводить та перевіряє один викладач, і кожна перевірка викладача стосується одного заходу поточного контролю, то між типами сутностей «Контроль викладача» та «Контроль модулю» існує тип зв’язку «один до одного», так само, як і між типами сутностей «Контроль викладача» та «Контроль теми».

Але за одним навчальним планом можуть навчатись декілька академічних груп. У цьому випадку для однієї дисципліни плану для одного екземпляру сутності «Контроль модулю» («Контроль теми») буде декілька екземплярів сутності «Контроль викладача» – окремий екземпляр на кожну групу. Таким чином, тип сутності «Контроль модулю» зв’язаний з типом сутності «Контроль викладача» типом зв’язку «один до багатьох»; так само і тип сутності «Контроль модулю» з типом сутності «Контроль викладача».

У схему потрібно додати новий базовий тип сутності «Група». Базовий тип сутності «Факультет» зв’язаний з типом сутності «Група» типом зв’язку «Один до багатьох». Кожна група навчається за деяким навчальним планом, при цьому декілька груп можуть навчатись за одним планом. Якщо розглядати у якості екземпляру сутності «Навчальний план» план одного семестру, то академічна груп у навчальному році навчається за двома планами – навчальним планом осіннього семестру, та навчальним планом весняного семестру. Отже, між типами сутностей «Група» та «Навчальний план» існує тип зв’язку «багато до багатьох». Тому додамо асоціативний тип сутності «План групи». Тоді між типом сутності «Група» та типом сутності «План групи» буде існує тип зв’язку «один до багатьох», так само, як між типом сутності «Навчальний план» та типом сутності «План групи».

Отримуємо остаточну концептуальну модель БД для підсхеми «Методична робота» (рис. 3.8).

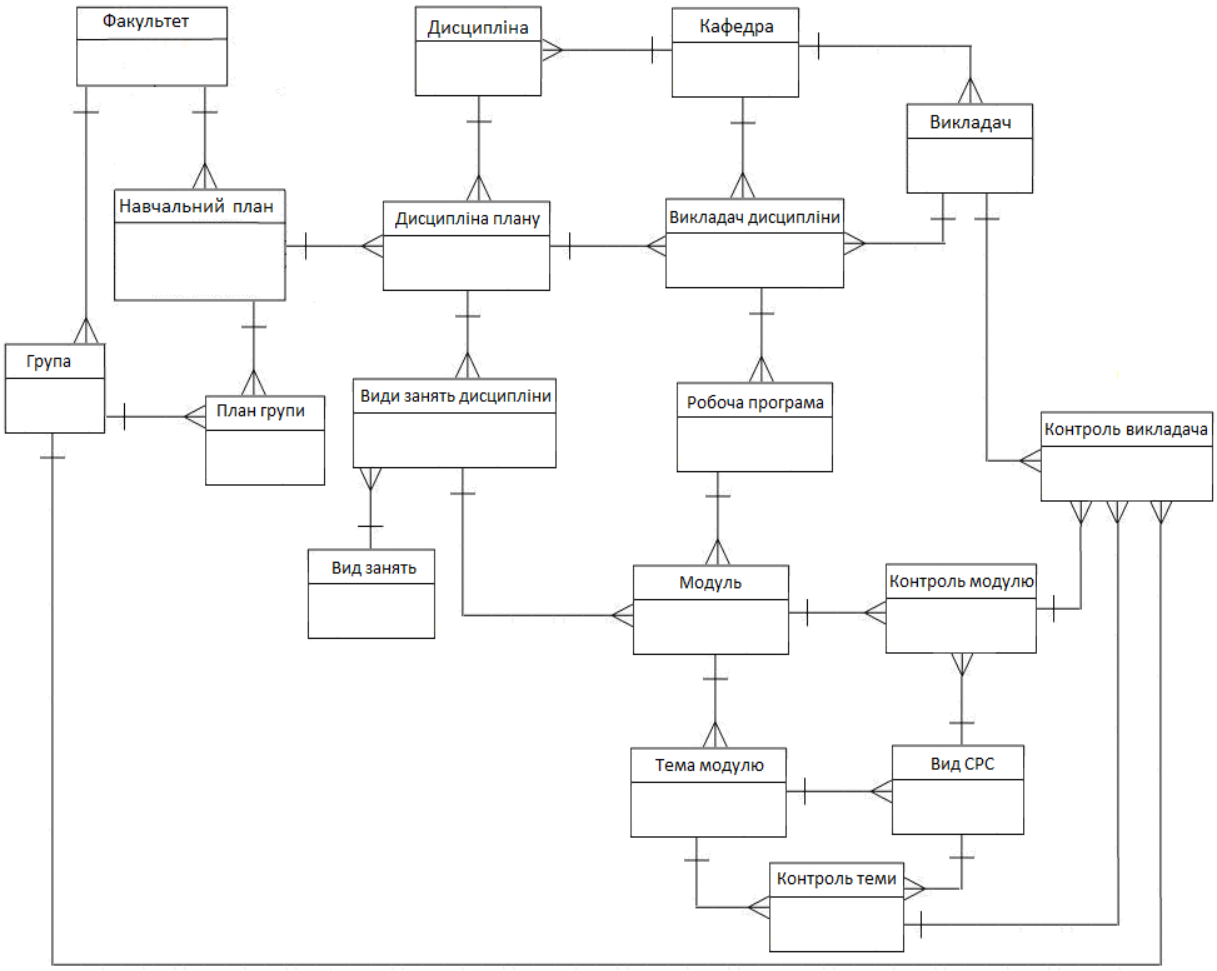

Рисунок 3.8 – Остаточна концептуальна модель бази даних для підсхеми «Методична робота»

3.2.2 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, семестр, кількість тижнів навчання.

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

У типу сутності «Дисципліна» будуть атрибути: код дисципліни (ідентифікатор), код кафедри, назва дисципліни, 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 «Дисципліна плану» будуть атрибути: код навчального плану, код дисципліни, кількість кредитів, вид семестрового контролю (іспит, залік), наявність та 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 занять дисципліни» будуть атрибути: код навчального плану, код дисципліни, код виду занять, кількість годин занять (за семестр).

У типу сутності «Модуль» будуть атрибути: назва, номер модулю, код навчального плану, код дисципліни, вид модулю (лекційний, практичний), максимальний бал (що може отримати студент у якості оцінки за виконання модулю).

У типу сутності «Вид СРС» будуть атрибути: код виду СРС, назва, кількість годин СРС, кількість годин перевірки.

У типу сутності «Контроль модулю» будуть атрибути: код виду СРС, номер модулю, код навчального плану, код дисципліни, процент від балу за модуль.

У типу сутності «Тема модулю» будуть атрибути: назва, номер модулю, номер теми, код навчального плану, код дисципліни, кількість годин аудиторних занять, кількість годин самостійної підготовки, бал за виконання.

У типу сутності «Контроль теми» будуть атрибути: код виду СРС, номер теми, номер модулю, код навчального плану, код дисципліни, процент від балу за тему.

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. Атрибут «кількість годин перевірки» відноситься до дійсного типу даних. Для цього атрибуту бажано призначати тип decimal. Вказані типи даних допустимі для СУБД, заснованих на будь-якої з сучасних моделей даних.

3.2.3 Визначення атрибутів, що є потенційними і первинними ключами

У всіх базових типів сутностей первинним ключем буде атрибут з назвою «ідентифікатор». У типів сутностей, що за змістом є таблицями-довідниками: «Факультет», «Кафедра», «Дисципліна», «Вид СРС», «Вид занять», «Вид методичної роботи», – 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. Для спрощення розрахунків у якості первинних ключів в цих таблицях можна призначати додаткові атрибути цілого типу, які будуть використовуватись у якості первинних ключів.

3.2.4 Визначення відповідності концептуальної моделі транзакціям користувачів

У розділі 3.2.1 перелічені транзакції групи користувачів «Працівник методичного відділу» ІС «Університет», що стосуються.

Розглянемо транзакції з точки зору можливості виконання їх.

1) Відновити нормативні документи щодо навчального процесу на наступний навчальний рік – ця робота методичного відділу не відображена у базі даних.

2) Отримати від уповноважених комісій ОПП та ОНП за спеціальностями – ця робота методичного відділу також не відображена у базі даних.
3) Затвердити ОПП та ОНП за спеціальностями – ця робота методичного відділу також не відображена у базі даних.
4) Отримати від факультетів робочі навчальні плани на наступний навчальний рік. Для цієї транзакції потрібно зробити вибірку даних з зв'язаних таблиць: «Факультет» – «Навчальний план» – «Дисципліна плану» – «Вид заняття дисципліни» – «Вид занять».
5) Затвердити робочі навчальні плани факультетів на наступний навчальний рік. Для цієї транзакції потрібно додати у перелік атрибутів типу сутності «Навчальний план» атрибут «Затверджено», логічного типу та призначити користувачеві – працівнику методичного відділу – права змінювати цей атрибут.
6) Скласти список дисциплін, що включені в навчальні плани на наступний рік та розподілити їх між кафедрами університету. Для цієї транзакції потрібно зробити вибірку даних з зв'язаних таблиць: «Факультет» – «Навчальний план» – «Дисципліна плану» – «Дисципліна» – «Кафедра».
7) Отримати від провідних викладачів робочі програми з дисциплін. Для цієї транзакції потрібно зробити вибірку даних з зв'язаних таблиць: «Кафедра» – «Дисципліна» – «Дисципліна плану» – «Викладач дисципліна» – «Модуль» – «Тема модулю» – «Контроль теми».
8) Затвердити робочі програми з дисциплін. Для цієї транзакції потрібно додати у перелік атрибутів типу сутності «Робоча програма» атрибут «Затверджено», логічного типу та призначити користувачеві – працівнику методичного відділу – права змінювати цей атрибут.
9) Відновити список можливих контролюючих заходів для оцінки успішності навчання студентів. Надати права користувачеві – працівнику методичного відділу – маніпулювання даними для типу сутності «Вид СРС».
10) Відновити перелік документів, що входять до НМК дисципліни. Потрібно додати до схеми БД тип сутності «НМК дисципліни» з атрибутами: ідентифікатор, назва, термін періодичного оновлення.

11) Відновити перспективні плани оновлення НМК по дисциплінах кафедри – ця робота методичного відділу також не відображена у базі даних.
12) Отримати від кафедр план розробки навчально-методичного забезпечення дисциплін на наступний навчальний рік – ця робота методичного відділу також не відображена у базі даних.
13) Відновити таблиці розрахунку методичного навантаження викладача. Потрібно додати у схему тип сутності «Методична робота» з атрибутами: ідентифікатор, назва, одиниця вимірювання роботи, коефіцієнт перерахунку одиниці роботи в академічну годину загального навантаження викладача. Працівникам методичного відділу надати права маніпулювання даними у цій таблиці.
14) Затвердити плани розробки навчально-методичного забезпечення дисциплін на наступний навчальний рік. Потрібно додати у схему тип сутності «Методична робота викладача» з атрибутами: ідентифікатор викладача, ідентифікатор методичної робота, назва роботи, запланована кількість одиниць вимірювання. Виконати вибірку зі зв’язаних таблиць: «кафедра» – «Викладач» – «Методична робота викладача» – «Методична робота». Правами маніпулювання даними в цій таблиці повинні володіти викладачі кафедр університету, в працівники методичного відділу матимуть тільки право вибірки є цієї таблиці.
15) Перевірити виконання викладачами заходів розробки навчально-методичного забезпечення дисциплін у попередньому навчальному році. Потрібно додати у тип сутності «Методична робота викладача» атрибут «Виконана кількість одиниць вимірювання». Виконати вибірку зі зв’язаних таблиць: «кафедра» – «Викладач» – «Методична робота викладача» – «Методична робота».

Після внесення вказаних нових типів сутностей: базового типу сутності «Методична робота» та асоціативного типу сутності «Методична робота викладача», та створення у БД таблиці допусків різних груп користувачів до інформації у БД, перелічені транзакції працівника методичного відділу можна буде виконати (рис. 3.9).

Для транзакцій, призначених групі користувачів «Працівник навчального відділу» визначені сутності, їх атрибути і зв’язки між ними.

Отже, отримана концептуальна модель коректна у контексті підсистеми «Навчальний відділ».

Для отримання логічної моделі бази даних інформаційної системи потрібно вибрати модель даних для БД, що проектується.

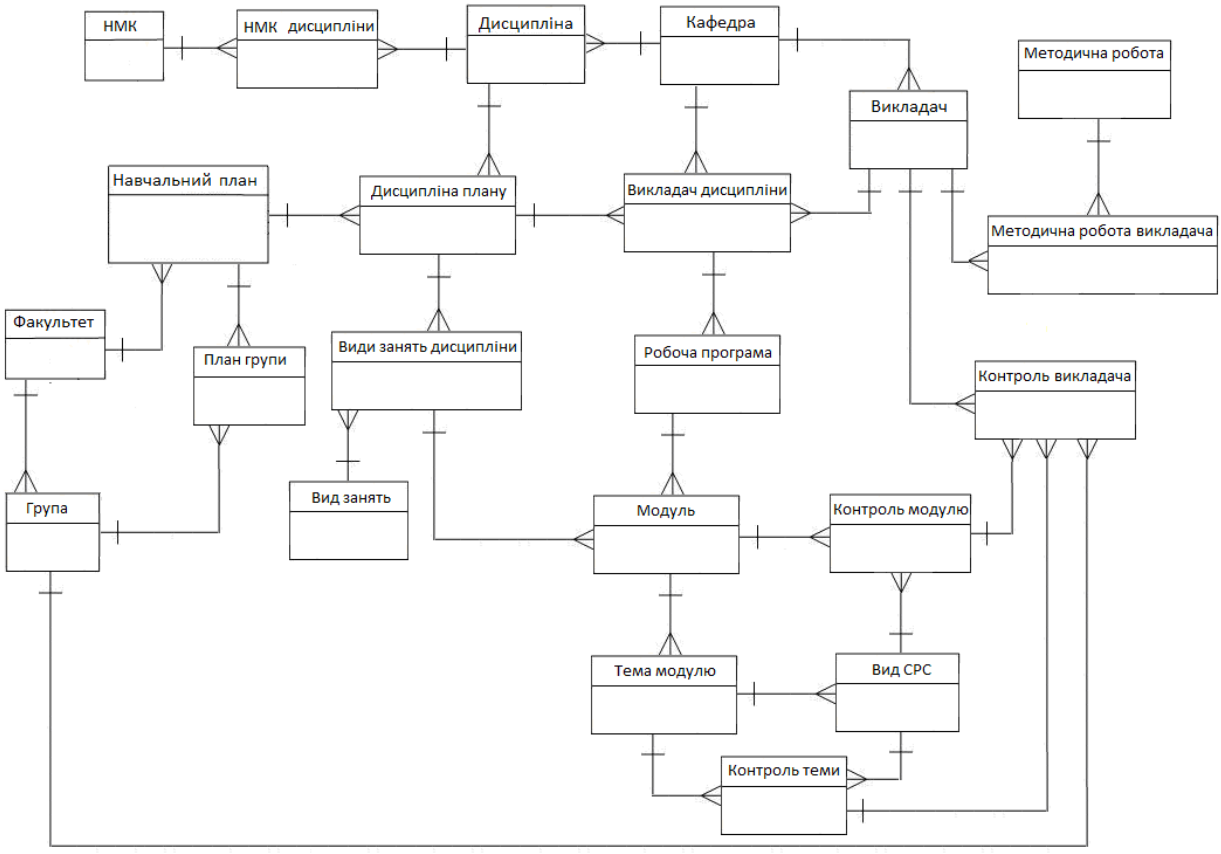

Рисунок 3.9 – Кінцева концептуальна модель БД схеми «Методична робота» для підсистеми «Методичний відділ»

Для отримання фізичної моделі бази даних потрібно також вибрати систему управління базами даних (СУБД), під управління якої буде працювати база даних.

4 ВИБІР СУБД ТА СЕРЕДОВИЩА РОЗРОБКИ ЗАСТОСУВАННЯ

4.1 Вибір СУБД

База даних ІС «Навчальний процес університету» розробляється поступово на протязі декількох років.

База даних була розроблена за реляційною моделлю. Для керування роботою бази даних була обрана СУБД компанії MicroSoft MS SQL Server. Спочатку БД була створена для версії СУБД MS SQL Server 2005, потім була перероблена для СУБД MS SQL Server 2008.

Для інформаційної системи поступово поповнюється серверне програмне забезпечення.

Для можливості використання попередніх розробок потрібно залишити для інформаційної системи СУБД MS SQL Server, можна лише вибрати більш нову версію. Але вибирати нову версію слід обачно, щоб не виникла потреба переробки серверного ПЗ, або зміни структури таблиць, наприклад, типів даних атрибутів таблиць. Обирати іншу СУБД необачно, оскільки це призведе до необхідності переробки програмного коду, але не надасть ніяких додаткових можливостей [5].

MS SQL Server 2008 є безкоштовним випуском SQL Server і являє собою ідеальну платформу даних для навчання і створення невеликих серверних додатків, які можуть поширюватися незалежними постачальниками програмного забезпечення. Мова SQL у версії MS SQL Server та процедура мова T-SQL є зручними засобами для розробки серверного програмного забезпечення [6 –9].

Тому на цей час можна залишити СУБД версії MS SQL Server 2008.

4.2 Вибір середовища розробки застосування та мови програмування

Ніяке професійне застосування не може обійтися без зберігання даних в сторонніх сховищах. Тому доступ до зовнішніх джерел даних і їх зберігання в додатку є однією з найістотніших прикладних проблем, що вирішуються при створенні додатків

Для розробки додатків до баз даних звичайно використовуються середовища швидкої розробки застосувань. Раніше лідером у розробці середовищ швидкої розробки програмних застосувань була компанія Borland, але у останні роки лідерство перейшло до компанії MicroSoft з її продуктом Visual Studio.

У наш час продукти Microsoft для розробників входять до списку найбільш затребуваного програмного забезпечення для розробників додатків до баз даних [8 – 11].

Microsoft Visual Studio – лінійка продуктів компанії Майкрософт, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Microsoft Silverlight.

Продукт компанії Microsoft – Visual Studio 2015 – є потужним середовищем розробки, що забезпечує високу якість коду на протязі всього циклу життя програмного забезпечення, від проектування до впровадження.

Інтегроване середовище розробки (IntegratedDevelopmentEnvironment – IDE) Visual Studio пропонує ряд високорівневих функціональних можливостей, які виходять за рамки базового управління кодом.

Однією з багатьох можливостей є підтримка множини мов при розробці. Visual Studio дозволяє писати код своєю мовою чи будь-якою іншою бажаною мовою, використовуючи весь час один і той же інтерфейс (IDE). Більш того, Visual Studio також ще дозволяє створювати Web-сторінки на різних мовах, але поміщати їх все в один і той же Web-додаток. Єдиним обмеженням є те, що в кожної Web-сторінці можна використовувати тільки якийсь один мову (очевидно, що в іншому випадку проблем при компіляції було б просто не уникнути).

Microsoft Visual Studio 2015 – це потужне середовище швидкої розробки програмних застосувань різного виду, якому властиві всі переваги минулих версій цього середовища розробки, та у якому додані нові можливості. Основні зміни, які відбулися в порівнянні з попередніми версіями, можна віднести до таких груп, як ефективність, поліпшення існуючих технологій, розширюваність, вплив поточних тенденцій.

Visual Studio 2015 оснащена всіма удосконаленими засобами тестування для забезпечення стабільної якості коду. Розробники застосувань можуть використовувати кодовані тести інтерфейсу користувача, за допомогою яких можна автоматизувати тестування інтерфейсу веб-додатків і

додатків Windows, засоби ручного тестування, функцію Test Professional і інші корисні функції.

Модульне тестування та розробка, керована тестами – середовище розробки Visual Studio 2015 включає в себе велику кількість розширень, які допомагають в реалізації цих стратегій.

Розробка баз даних вимагає тієї ж ретельності і уваги що і розробка додатків. У Visual Studio 2015 це враховується: користувачам надаються надійні засоби розробки та управління змінами, які дозволяють синхронізувати додаток і базу даних.

4.2.1 Мова програмування C#

На сьогоднішній момент мова програмування C# є однією з найпотужніших і затребуваних мов в IT-галузі, і ця мова швидко розвиваються. На даний момент мовою C# пишуться найрізноманітніші програми: від невеликих десктопних програмок до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів.

У порівнянні з іншими мовами C# досить молода, але в той же час вона вже пройшла великий шлях. Перша версія мови вийшла разом з релізом Microsoft Visual Studio .NET в лютому 2002 року. Поточною версією мови є версія C# 7.0, яка вийшла в 7 березня 2017 року разом з Visual Studio 2017.

C# є мовою з Сі-подібним синтаксисом і близька в цьому відношенні до C++ і Java. Тому C# легко вивчати програмістам, що знайомі з однією з вказаних мов.

Мова C# є об’єктно-орієнтованою і в цьому плані багато перейняла у мов програмування Java і C++. Наприклад, C# підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об’єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. Мова C# продовжує активно розвиватися, і з кожною новою версією з'являється все більше цікавих функціональних можливостей, як, наприклад, лямбда-вирази, динамічне зв’язування, асинхронні методи і т.д.

Коли говорять C#, нерідко мають на увазі технології платформи .NET (WPF, ASP.NET). І, навпаки, коли говорять .NET, нерідко мають на увазі C#. Однак, хоча ці поняття пов’язані, ототожнювати їх невірно. Мова C# була створена спеціально для роботи з фреймворком .NET, проте саме поняття .NET дещо ширше.

Фреймворк .NET представляє потужну платформу для створення додатків. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов: поряд з C# це також VB.NET, C++, F#, а також різні діалекти інших мов, прив'язані до .NET, наприклад, Delphi.NET. При компіляції код на будь-якій з цих мов компілюється в збірку спільною мовою CIL (Common Intermediate Language) – свого роду асемблер платформи .NET. Тому можна розробляти окремі модулі однієї програми на окремих мовах.

.NET Framework є платформою, яка підтримується на більшості сучасних ОС Windows (Windows 10/8.1/8.7/Vista), а також можна створювати додатки, що будуть працювати на мобільних платформах Android і iOS. .NET представляє єдину для всіх підтримуваних мов бібліотеку класів. І який би додаток не треба було написати на C#, так чи інакше потрібно задіяти бібліотеку класів .NET.

Загальномовне середовище виконання CLR і базова бібліотека класів є основою для цілого стека технологій, які розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних в цьому стеку технологій призначена технологія ADO.NET.

Таким чином програмне середовище розробки застосувань компанії Microsoft Visual Studio 2015 та мова C# є дуже вдалим вибором у якості середовища розробки застосувань для баз даних.

5 ЛОГІЧНЕ І ФІЗИЧНЕ ПРОЕКТУВАННЯ БД

У третьому розділі була отримана концептуальна модель бази даних підсистеми «Методичний відділ». Цю модель потрібно перетворити у логічну модель БД у відповідності до обраної моделі даних.

У четвертому розділі в якості моделі даних була вибрана реляційна модель. Тому логічне проектування необхідно виконати для цієї моделі.

5.1 Логічне проектування

Розроблена в третьому розділі модель вже є логічною моделлю реляційної бази даних. Як видно з концептуальної моделі, представленої на рис.3.8, ніде немає типів зв’язків «багато до багатьох», які не допустимі у реляційної бази даних.

Таким чином, набір сутностей, визначених при концептуальному проектуванні, перетворюється в набір відношень при логічному проектуванні по реляційній моделі. Після усунення з моделі зв’язків типу «багато до багатьох» відношення задовольняють правилам нормалізації.

Перевірку відповідності відношень вимогам призначених для користувача транзакцій робити не має сенсу, оскільки логічна модель 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. Як видно з рис. 3.8, діаграма бази даних суттєво відрізняється від діаграми у вигляді ієрархічного дерева. У цих випадках потрібно намагатись уникати встановлення для зовнішніх ключів можливості каскадного оновлення або усунення записів у підлеглих відношеннях.

Таким чином, отримана логічна модель бази даних. Логічна схема БД – підсхема «Навчальний відділ» – приведена в додатку А.

5.2 Фізичне проектування

При виконанні фізичного проектування, як вказувалося в третьому 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 видів СРС.

В інформаційній системі, що розробляється, передбачається велика кількість користувачів з правами маніпулювання даними в таблицях бази даних. У подібних випадках небажано мати первинні ключі з властивістю автоматичної нумерації для таблиць, права на додавання і видалення даних в яких мають багато користувачів.

Фізична схема БД – підсхема «Методичний відділ» – приведена в додатку Б.

6 СЕРВЕРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

В інформаційній системі «Навчальний процес університету», підсистемою якого є ІС «Методичний відділ», передбачається наявність декілька груп користувачів, які мають права маніпулювати даними в БД за допомогою офісних застосувань.

У цих випадках бажано розробляти клієнтські застосування бажано у вигляді «тонких клієнтів», щоб застосування лише надавали користувачам необхідний інтерфейс, а обробка даних відбувалася на боці серверу.

Для тонких клієнтів бажано виконувати команди маніпулювання даними у таблицях БД через виклик збережених процедур бази даних. А вже збережені процедури будуть виконувати необхідні дії по маніпулюванню даними у таблицях БД. У клієнтському застосуванні після виклику збережених процедур, що змінюють вміст таблиць БД потрібно лише оновити компоненти форми програми, що відображають інформацію у таблицях.

Процедура CPCAdd додає запис у таблицю KindCPC:

```
CREATE procedure CPCAdd
  @id_cpc smallint output,
  @name_cpc varchar (60),
  @st_hour tinyint,
  @teach_hour decimal (4,2),
  @k tinyint output,
  @msg varchar (50) output
```

Параметри процедури:

@id_cpc – ідентифікатор виду СРС;

@name_cpc – назва виду СРС;

@st_hour – кількість годин, відведених студенту на виконання даного виду СРС;

@teach_hour – кількість годин, відведених викладачеві на перевірку виконання академічного групою контролюючого заходу для даного виду СРС;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура CPCAdd перевіряє, чи вже є в БД запис про вид СРС з заданою назвою. Якщо такий запис є, то генерується повідомлення про помилку.

Якщо такого запису немає, то в таблицю KindCPC додається запис зі значеннями атрибутів: name_cpc = @name_cpc, teach_hour = @teach_hour, st_hour = @st_hour. Процедура визначає для доданого запису значення первинного ключа таблиці KindCPC– ідентифікатор @id_cpc, – та повертає його значення.

Процедура CPCEdit додає запис у таблицю KindCPC:

```
CREATE procedure CPCEdit
  @id_cpc smallint,
  @name_cpc varchar (60),
  @st_hour tinyint,
  @teach_hour decimal (4,2),
  @k tinyint output,
  @msg varchar (50) output
```

Параметри процедури:

@id_cpc – ідентифікатор виду СРС;

@name_cpc – назва виду СРС;

@st_hour – кількість годин, відведених студенту на виконання даного виду СРС;

@teach_hour – кількість годин, відведених викладачеві на перевірку виконання академічного групою контролюючого заходу для даного виду СРС;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура CPCEdit перевіряє, чи вже є в БД запис про вид СРС з заданою назвою іншим значенням ідентифікатору id_cpc. Якщо такий запис є, то генерується повідомлення про помилку.

Якщо такого запису немає, то в таблиці KindCPC оновлюється з ключем id_cpc = @id_cpc. Для нього встановлюються значення атрибутів: name_cpc = @name_cpc, teach_hour = @teach_hour, st_hour = @st_hour.

Процедура CPCDel усуває запис з таблиці KindCPC:

```
CREATE  procedure CPCDel
  @id_cpc smallint,
  @m bit,
  @k tinyint output,
  @msg varchar (50) output
```

Параметри процедури:

@id_cpc – ідентифікатор аудиторії;

@m – прапор, чи видаляти записи з підлеглих таблиць;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Якщо @m=0, то процедура CPCDel перевіряє, чи є у підлеглих таблицях записи, що зв'язані по ключу id_cpc з записом таблиці KindCPC, що видаляється. Якщо такі записи є, то генерується повідомлення про помилку. Якщо таких записів немає, то запис з ключом id_cpc = @id_cpc видаляється з таблиці Auditory.

Якщо @m=1, то процедура CPCDel видаляє з таблиці KindCPC запис з вказаним значенням первинного ключа, а також з підлеглих таблиць видаляються записи, з відповідним значенням зовнішнього ключа.

Процедура NMKAdd додає запис у таблицю KindNMK:

```
CREATE procedure NMKAdd
  @id_nmk smallint output,
  @name_nmk varchar (60),
  @amountYear smallint,
  @k tinyint output,
  @msg varchar (50) output
```

Параметри процедури:

@id_nmk – ідентифікатор пункту НМК;

@name_nmk – назва пункту НМК;

@amountYear – кількість років, на протязі яких даний вид методичного забезпечення дисципліни можна не оновлювати;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура NMKAdd перевіряє, чи вже є в БД запис про вид СРС з заданою назвою. Якщо такий запис є, то генерується повідомлення про помилку.

Якщо такого запису немає, то в таблицю KindNMK додається запис зі значеннями атрибутів: name_nmk = @name_nmk, amountYear = @amountYear. Процедура визначає для доданого запису значення первинного ключа таблиці KindNMK – ідентифікатор @id_nmk, – та повертає його значення.

Процедура NMKEdit додає запис у таблицю KindNMK:

```
CREATE procedure NMKEdit
  @id_nmk smallint,
  @name_nmk varchar (60),
  @amountYear smallint,
```

@k tinyint output,

@msg varchar (50) output

Параметри процедури:

@id_nmk – ідентифікатор пункту НМК;

@name_nmk – назва пункту НМК;

@amountYear – кількість років, на протязі яких даний вид методичного забезпечення дисципліни можна не оновлювати;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура NMKEdit перевіряє, чи вже є в БД запис про пункт НМК з заданою назвою. Якщо такий запис є, то генерується повідомлення про помилку.

Якщо такого запису немає, то в таблиці KindNMK оновлюється з ключем id_nmk = @id_nmk. Для нього встановлюються значення атрибутів: name_nmk = @name_nmk, amountYear = @amountYear.

Процедура NMKDel усуває запис з таблиці KindNMK:

CREATE procedure NMKDel

@id_nmk smallint,

@m bit,

@k tinyint output,

@msg varchar (50) output

Параметри процедури:

@id_nmk – ідентифікатор пункту НМК;

@m – прапор, чи видаляти записи з підлеглих таблиць;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Якщо @m=0, то процедура NMKDel перевіряє, чи є у підлеглих таблицях записи, що зв’язані по ключу id_nmk з записом таблиці KindNMK, який видаляється. Якщо такі записи є, то генерується повідомлення про помилку. Якщо таких записів немає, то запис з ключом id_nmk = @id_nmk видаляється з таблиці KindNMK.

Якщо @m=1, то процедура NMKDel видаляє з таблиці KindNMK запис з вказаним значенням первинного ключа, а також з підлеглих таблиць видаляються записи, з відповідним значенням зовнішнього ключа.

Процедура MetodWAdd додає запис у таблицю MetodWork:

CREATE procedure MetodWAdd

@id_mw smallint output,

@name_mw varchar (60),
@short_mw varchar (20),
@unit_mw varchar (20),
@coeff_hour decimal (5,2),
@status bit,
@k tinyint output,
@msg varchar (50) output

Параметри процедури:

@id_mw – ідентифікатор виду методичної роботи;

@name_mw– назва виду методичної роботи;

@short_mw – скорочена назва виду методичної роботи;

@unit_mw – одиниця вимірювання методичної роботи;

@coeff_hour – коефіцієнт перерахування одиниці вимірювання методичної роботи у академічні години навантаження;

@status – чи може цю роботу виконувати тільки провідний викладач;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура MetodWAdd перевіряє, чи вже є в БД в таблиці MetodWork запис про вид методичної роботи з заданою назвою. Якщо такий запис є, то генерується повідомлення про помилку.

Якщо такого запису немає, то в таблицю MetodWork додається запис зі значеннями атрибутів: name_mw = @name_mw, short_mw = @short_mw, unit_mw = @unit_mw, coeff_hour = @coeff_hour, status = @status.

Процедура визначає для доданого запису значення первинного ключа таблиці MetodWork – ідентифікатор @id_mw, – та повертає його значення.

Процедура MetodWEdit змінює запис у таблицю MetodWork:

CREATE procedure MetodWEdit
@id_mw smallint,
@name_mw varchar (60),
@short_mw varchar (20),
@unit_mw varchar (20),
@coeff_hour decimal (5,2),
@status bit,
@k tinyint output,
@msg varchar (50) output

Параметри процедури:

@id_mw – ідентифікатор виду методичної роботи;

@name_mw– назва виду методичної роботи;

@short_mw – скорочена назва виду методичної роботи;

@unit_mw – одиниця вимірювання методичної роботи;

@coeff_hour – коефіцієнт перерахування одиниці вимірювання методичної роботи у академічні години навантаження;

@status – чи може цю роботу виконувати тільки провідний викладач;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура MetodWEdit перевіряє, чи вже є в БД в таблиці MetodWork запис про вид методичної роботи з заданою назвою але з іншим ідентифікатором. Якщо такий запис є, то генерується повідомлення про помилку.

Якщо такого запису немає, то в таблиці MetodWork у записа з ідентифікатором id_mw = @id_mw змінюється значеннями атрибутів: name_mw = @name_mw, short_mw = @short_mw, unit_mw = @unit_mw, coeff_hour = @coeff_hour, status = @status.

Процедура MetodWDel усуває запис з таблиці MetodWork:

```
CREATE  procedure MetodWDel
  @id_mw smallint,
  @m bit,
  @k tinyint output,
  @msg varchar (50) output
```

Параметри процедури:

@id_nmk – ідентифікатор виду методичної роботи;

@m – прапор, чи видаляти записи з підлеглих таблиць;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Якщо @m=0, то процедура MetodWDel перевіряє, чи є у підлеглих таблицях записи, що зв’язані по ключу id_mw з записом таблиці MetodWork, який видаляється. Якщо такі записи є, то генерується повідомлення про помилку. Якщо таких записів немає, то запис з ключом id_mw = @id_mw видаляється з таблиці MetodWork.

Якщо @m=1, то процедура NMKDel видаляє з таблиці KindNMK запис з вказаним значенням первинного ключа, а також з підлеглих таблиць видаляються записи, з відповідним значенням зовнішнього ключа.

7 ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ

7.1 Загальні відомості

Програма Metodical.exe розроблена у середовище швидкої розробки застосувань Visual Studio 2015 [10 – 15].

Програма написана на мові програмування C#. Програма є додатком до бази даних StudyingProccess, тому вона може працювати у локальній мережі, де розташований сервер баз даних з завантаженою БД StudyingProccess.

7.2 Функціональне призначення

Програма Metodical.exe розроблена у якості офісного додатку до бази даних StudyingProccess. Програма розроблена як автоматизоване робоче місце працівника методичного відділу університету.

Програма призначена для перегляду в зручному вигляді даних, що використовуються для методичного забезпечення навчального процесу університету. Також застосування дозволяє оновлювати ті дані, на маніпулювання якими користувач має право.

7.3 Керівництво програміста

Програма розроблена як проект виду WindowsFormApplication. Проект має одну форму. Програма включає три класи: Program, MainForm, List_Item.

Клас List_Item, використовується для заповнення випадних списків на формі програми результатами вибірки з таблиць бази даних. Цей клас не має конструктору та методів.

Клас Form1 – клас форми програми. Конструктор класу не має параметрів:

public Form1()

Конструктор класу Form1 ініціалізує компоненти форми, заповнює компоненти форми програми, що є випадними списками: depComboBox, depComboBox2, depComboBox3, depComboBox4, – списками кафедр університету, які отримує як рядки запиту до БД.

Також заповнює випадний список facComboBox переліком факультетів університету, які отримує як рядки запиту до БД.

Конструктор форми ініціалізує об’єкти, що відображають збережені процедури бази даних. Далі програма працює у режимі опрацювання подій.

Методи класу форми Form1.

Метод init_commands_delCPCCmd призначений для ініціалізації команди виклику збереженої процедури CPCDel:

private void init_commands_delCPCCmd()

Метод init_commands_addCPCCmd призначений для ініціалізації команди виклику збереженої процедури CPCAdd:

private void init_commands_addCPCCmd ()

Метод init_commands_editCPCCmd Призначений для ініціалізації команди виклику збереженої процедури CPCEdit:

private void init_commands_editCPCCmd ()

Метод init_commands_delNMKCmd призначений для ініціалізації команди виклику збереженої процедури NMKDel:

private void init_commands_delNMKCmd()

Метод init_commands_addNMKCmd призначений для ініціалізації команди виклику збереженої процедури NMKAdd:

private void init_commands_addNMKCmd ()

Метод init_commands_editNMKCmd призначений для ініціалізації команди виклику збереженої процедури NMKEdit:

private void init_commands_editNMKCmd ()

Метод init_commands_delMWCmd призначений для ініціалізації команди виклику збереженої процедури MetodWDel:

private void init_commands_delMWCmd()

Метод init_commands_addMWCmd призначений для ініціалізації команди виклику збереженої процедури MetodWAdd:

private void init_commands_addMWCmd ()

Метод init_commands_editMWCmd призначений для ініціалізації команди виклику збереженої процедури MetodWEdit:

private void init_commands_editMWCmd ()

Метод fillDepComboBox заповнює випадні списки кафедр університету значеннями запиту до БД:

private void fillDepComboBox()

Метод fillDepComboBox отримує з запиту до БД список кафедр університету, та заповнює цим списком чотири компоненти типу ComboBox на вкладках форми: depComboBox, depComboBox2, depComboBox3, depComboBox4.

Метод fillFacComboBox заповнює випадний список факультетів університету значеннями запиту до БД:

private void fillFacComboBox ()

Метод fillFacComboBox отримує з запиту до БД список факультетів університету, та заповнює цим списком компонент типу ComboBox на вкладках форми: facComboBox.

Метод fillTeachComboBox заповнює випадний список викладачів заданої кафедри ЗВО:

private void fillTeachComboBox(short id_dep)

Параметр методу:

id_dep – ідентифікатор кафедри, де працюють викладачі.

Метод fillTeachComboBox отримує з запиту до БД список викладачів заданої кафедри: прізвище та ініціали, – та заповнює цим списком компоненти типу ComboBox teachComboBox1 та teachComboBox на останніх вкладках форми.

Метод fillSubComboBox заповнює випадний список дисциплін заданої кафедри університету:

private void fillSubComboBox (short id_dep)

Параметр методу:

id_dep – ідентифікатор кафедри.

Метод fillSubComboBox отримує з запиту до БД список скорочених назв дисциплін кафедри та заповнює цим списком компонент subComboBox на вкладці форми.

Решта методи класу Form1 являються обробниками подій натискання на кнопки або вибору пункту випадного списку.

Обробник події натискання на кнопку yearButton:

private void yearButton_Click(object sender, EventArgs e)

Метод yearButton_Click є обробником події натискання на кнопку «ОК» біля текстового поля для вводу номеру навчального року. При натисканні на цю кнопку з рядку тексту у випадному списку вибирається значення для змінної LearnYear, що використовується як параметр у запитах до БД.

Обробник події вибору пункту випадного списку depComboBox:

private void depComboBox_SelectedIndexChanged(object sender,
    EventArgs e)

private void depComboBox2_SelectedIndexChanged(object sender,
    EventArgs e)

private void depComboBox3_SelectedIndexChanged(object sender,
    EventArgs e)

private void depComboBox4_SelectedIndexChanged(object sender,
EventArgs e)

Перелічені методи є обробниками події вибору пункту випадного списку «Кафедра». Цей метод зчитує ідентифікатор вибраної кафедри та записує у змінну id_dep. Змінна id_dep використовується як параметр запитів до БД, які виводять дані по кафедрі на компоненти типу DataGridView на формі програми. Також змінна id_dep є параметром запиту, за допомогою якого випадні список subComboBox, teachComboBox та teachComboBox1 назвами дискиін кафедри або прізвищами викладачів вибраної кафедри.

Обробник події вибору пункту випадного списку facComboBox:

private void facComboBox_SelectedIndexChanged(object sender,
EventArgs e)

Метод є обробником події вибору пункту випадного списку «Факультет». Цей метод зчитує ідентифікатор вибраного факультету та записує у змінну id_f. Змінна id_f використовується як параметр запитів до БД, які виводять дані по факультету на компонент типу DataGridView на формі програми.

Обробник події вибору пункту випадного списку teachComboBox:

private void teachComboBox_SelectedIndexChanged(object sender,
EventArgs e)

Метод є обробником події вибору пункту випадного списку викладачів кафедри. Метод teachComboBox_SelectedIndexChanged зчитує в змінну id_teach ідентифікатор вибраного у списку викладача.

Обробник події натискання на кнопку addNMKButton:

private void addNMKButton_Click(object sender, EventArgs e)

Метод addNMKButton_Click є обробником події натискання на кнопку «Добавить пункт НМК» на вкладці форми «НМК дисциплин». Цей метод робить недоступними кнопки на сторінці та робить видимими приховані елементи форми, які використовуються для введення даних при додаванні запису у таблицю.

Обробник події натискання на кнопку editNMKButton:

private void editNMKButton_Click(object sender, EventArgs e)

Метод editNMKButton_Click є обробником події натискання на кнопку «Изменить пункт НМК» на вкладці «НМК дисциплин». Цей метод також робить недоступними кнопки на сторінці та робить видимими приховані елементи форми, які використовуються для оновлення запису.

Обробник події натискання на кнопку delNMKButton:

private void delNMKButton_Click(object sender, EventArgs e)

Цей метод є обробником події натискання на кнопку «Удалить пункт НМК» також на вкладці «НМК дисциплин». Метод виводить на форму запитання, чи дійсно користувач хоче видалити з бази даних запис про вибраний пункт НМК. Якщо користувач вибирає відповідь «ОК», то викликається збережена процедура бази даних NMKDel для видалення запису в таблиці KindNMK.

Обробники події зміни поточного рядка у компоненті dataGridView1:

private void dataGridView1_RowEnter(object sender,

DataGridViewCellEventArgs e)

Цей метод зчитує в змінну id_plan значення ідентифікатору навчального плану у поточному рядку табличного компоненту на вкладці формі dataGridView1. Змінна id_plan є параметром запиту, який вибирає дані для заповнення другого табличного компоненту dataGridView2.

Обробники події зміни поточного рядка у компоненті dataGridView7:

private dataGridView7_RowEnter (object sender,

DataGridViewCellEventArgs e)

Цей метод зчитує в змінні id_plan, id_mod відповіднізначення ідентифікаторів у поточному рядку табличного компоненту dataGridView7. Ці змінні є параметрами запиту, який вибирає дані для заповнення інших табличних компонентів на цій вкладці.

7.4 Посібник користувача

При запуску програми Metodical.exe на екрані з’являється форма програми (рис. 7.1). Спочатку користувачеві потрібно вибрати навчальний рік, з даними по якому він збирається працювати.

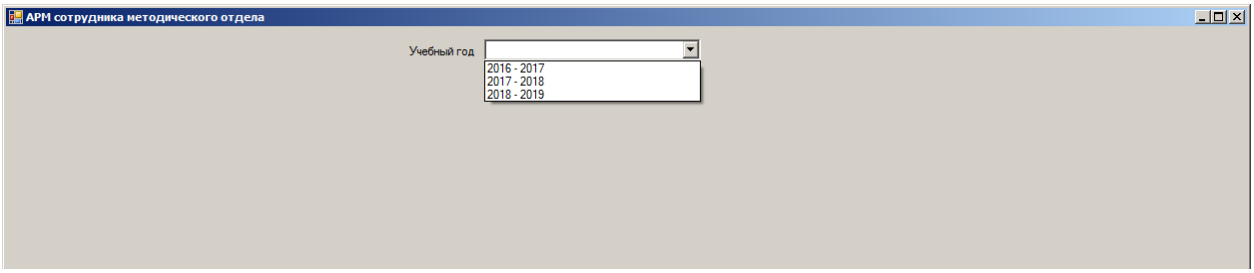

Рисунок 7.1 – Початок роботи з програмою

Після вибору навчального року на формі програми стає видимим компонент з багатьма вкладками, на які виводяться дані.

Перша вкладка призначена для перегляду та корегування списку документів, що складають НМК дисципліни (рис. 7.2).

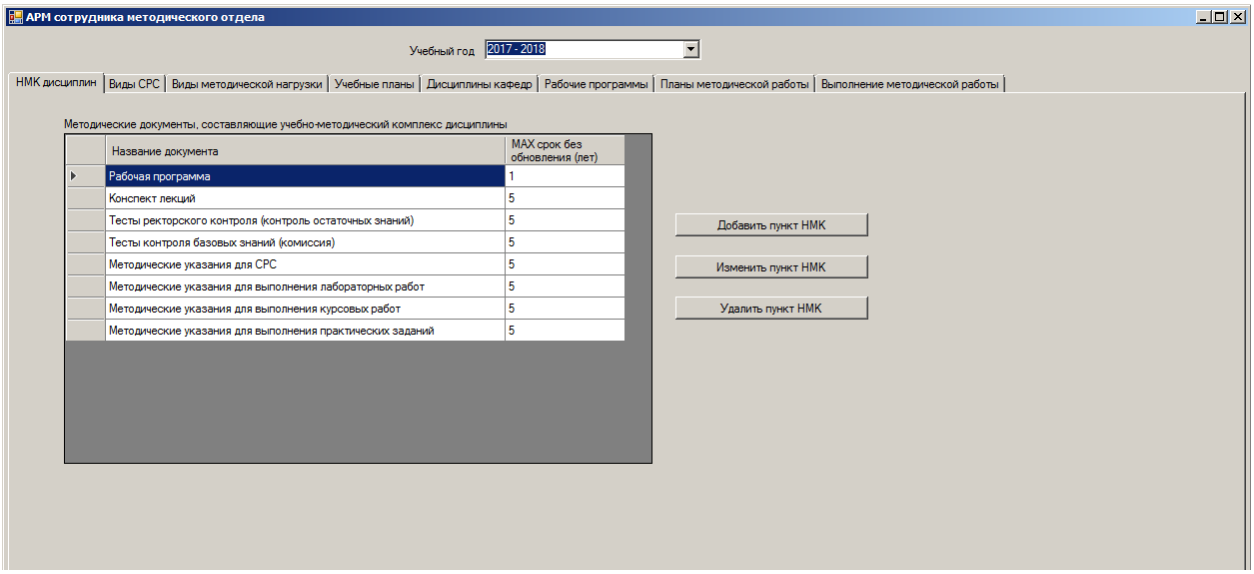

Рисунок 7.2 – Вкладка «НМК дисциплин»

На вкладці є кнопки для додавання, оновлення та усунення записів записів з таблиці переліку документів НМК дисципліни. При натисканні на кнопку «Добавить пункт НМК» на формі стають видимими компоненти для додавання запису у таблицю (рис. 7.3).

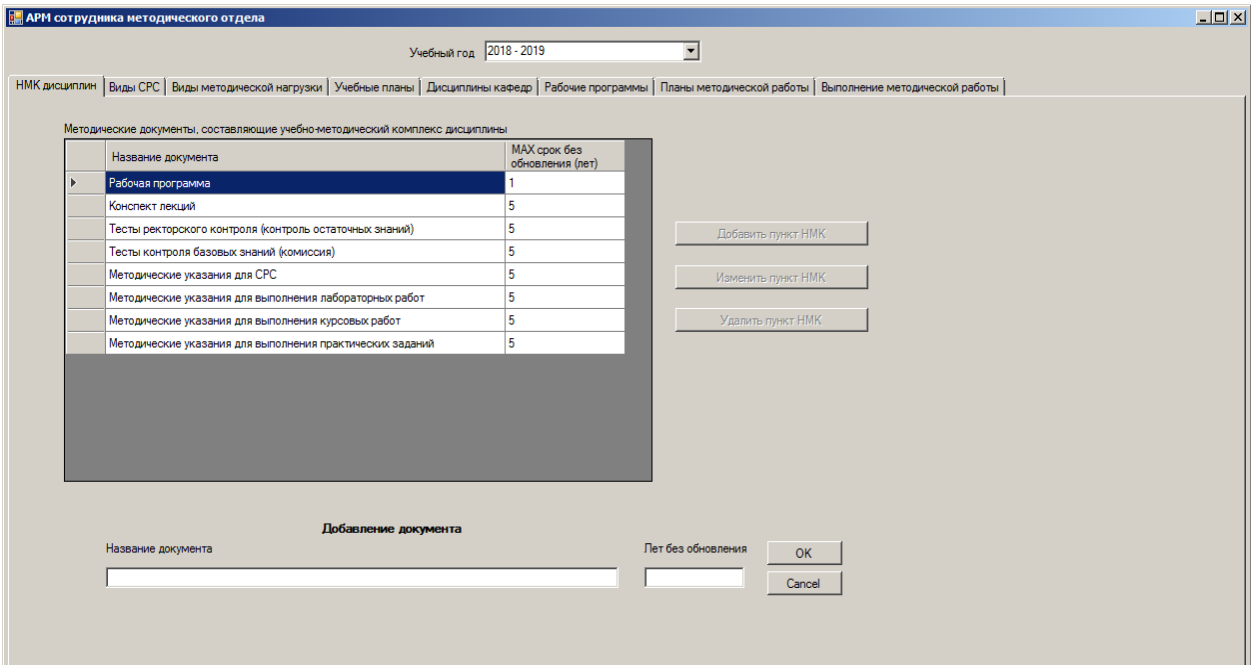

Рисунок 7.3 – Елементи форми для додавання запису у таблицю

При натисканні на кнопку «Изменить пункт НМК» на формі стають видимими компоненти ті самі компоненти вже заповнені даними з поточного рядка табличного компоненту (рис. 7.4).

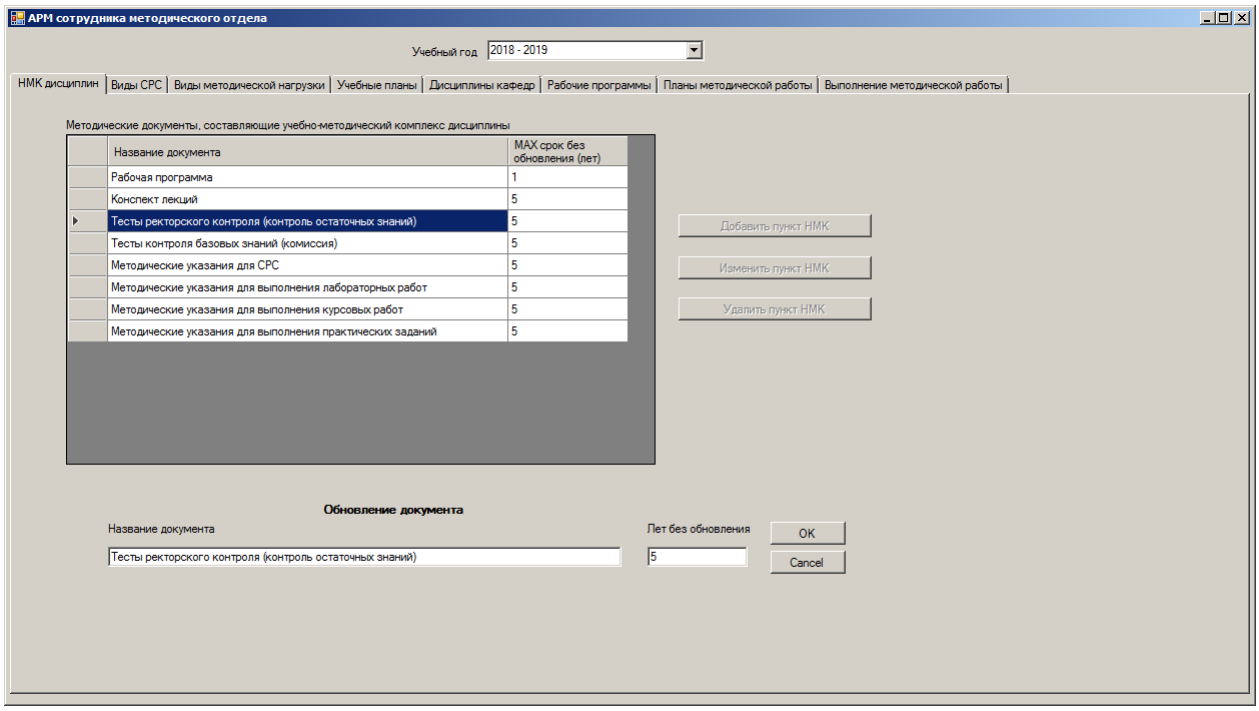

Рисунок 7.4 – Можливість оновлення запису у таблиці

Друга вкладка призначена для перегляду та корегування списку контролюючих заходів СРС (рис. 7.5).

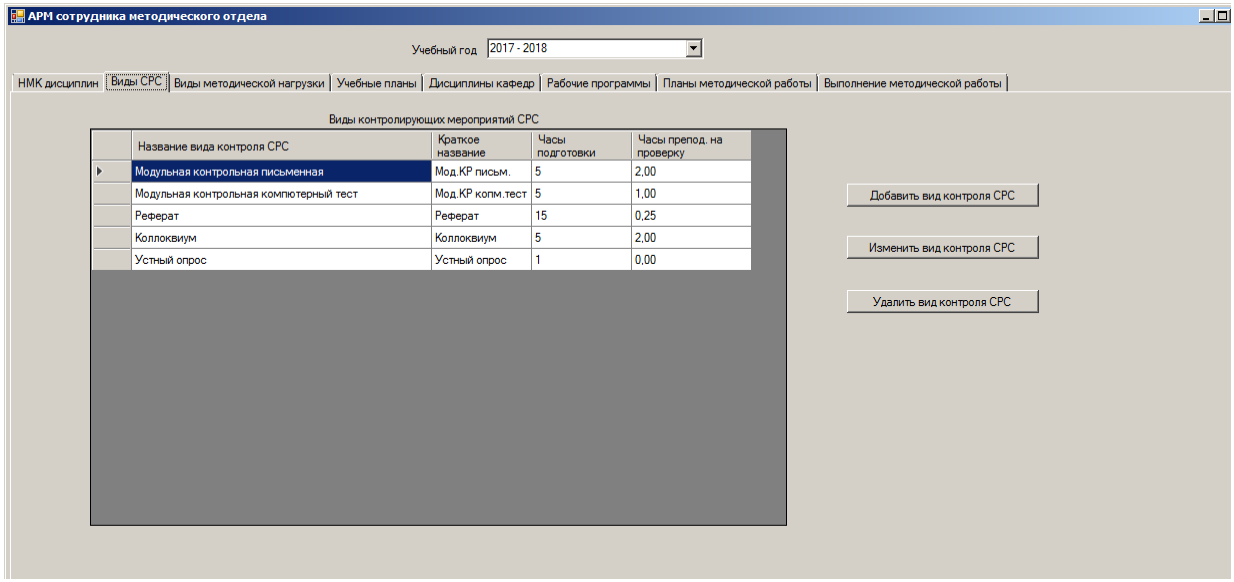

Рисунок 7.5 – Вкладка «Виды СРС»

На цій вкладці також є кнопки для маніпулювання даними в таблиці. При натисканні відповідних кнопок стають видимими компоненти для додавання та оновлення запису в таблиці (рис. 7.6 – 7.7).

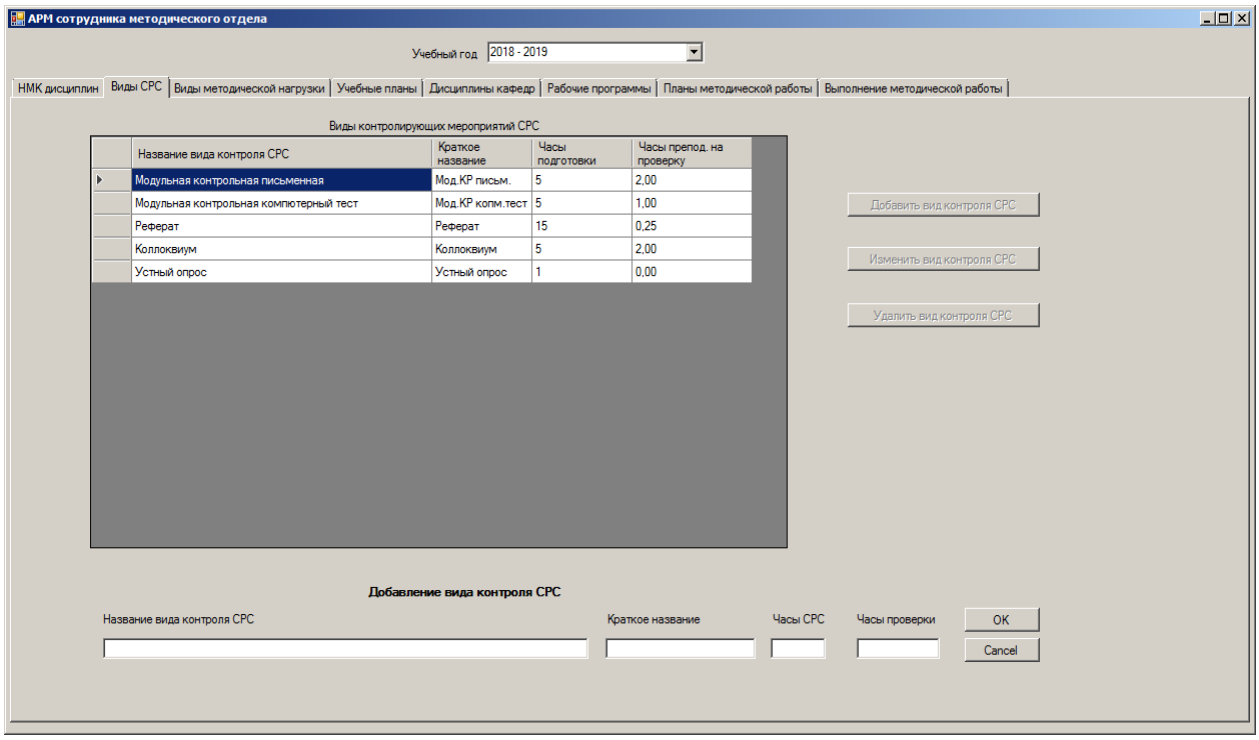

Рисунок 7.6 – Можливість додавання запису у таблицю видів СРС

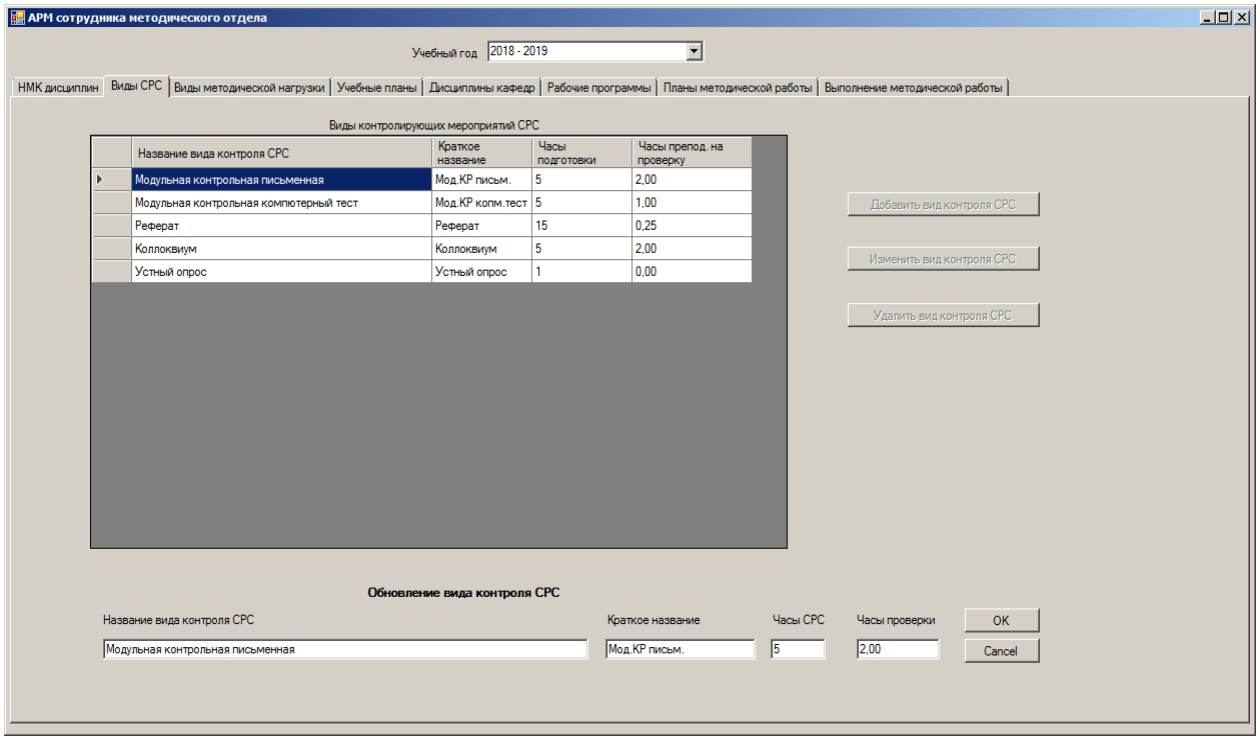

Рисунок 7.7 – Можливість оновлення запису у таблиці видів СРС

Третя вкладка призначена для перегляду та корегування списку видів методичної роботи викладачів (рис. 7.8). На цій вкладці також є кнопки для маніпулювання записами в таблиці видів методичної роботи (рис. 7.9 – 7.10).

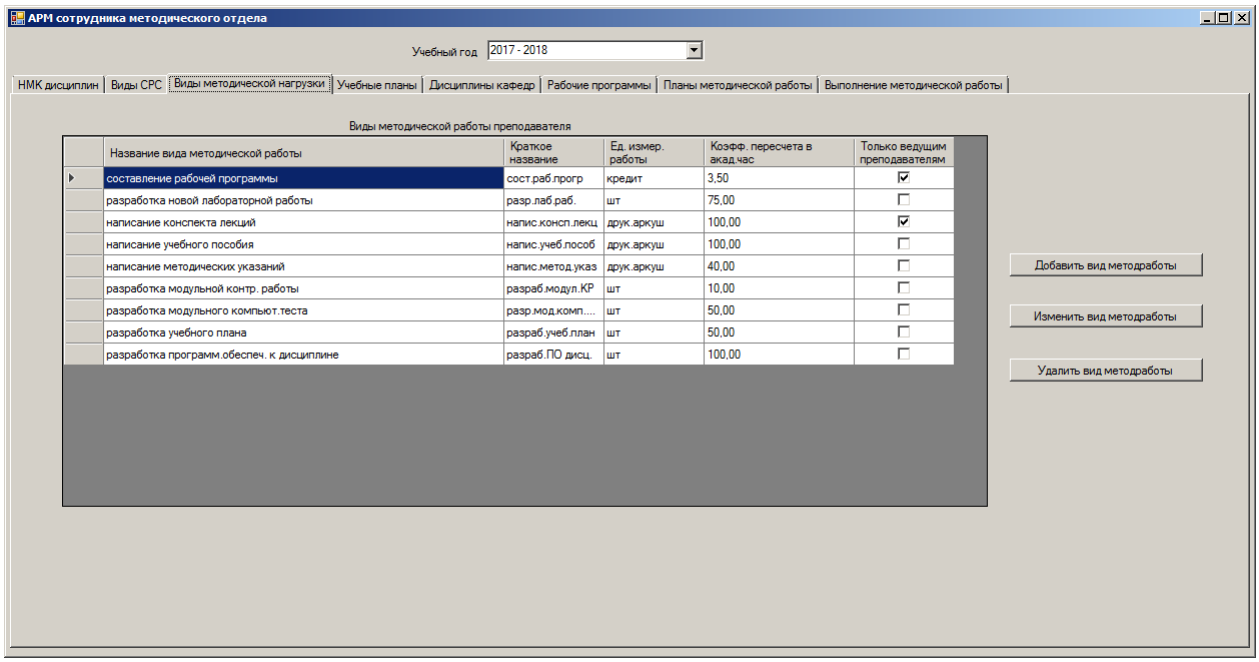

Рисунок 7.8 – Вкладка «Виды методической работы»

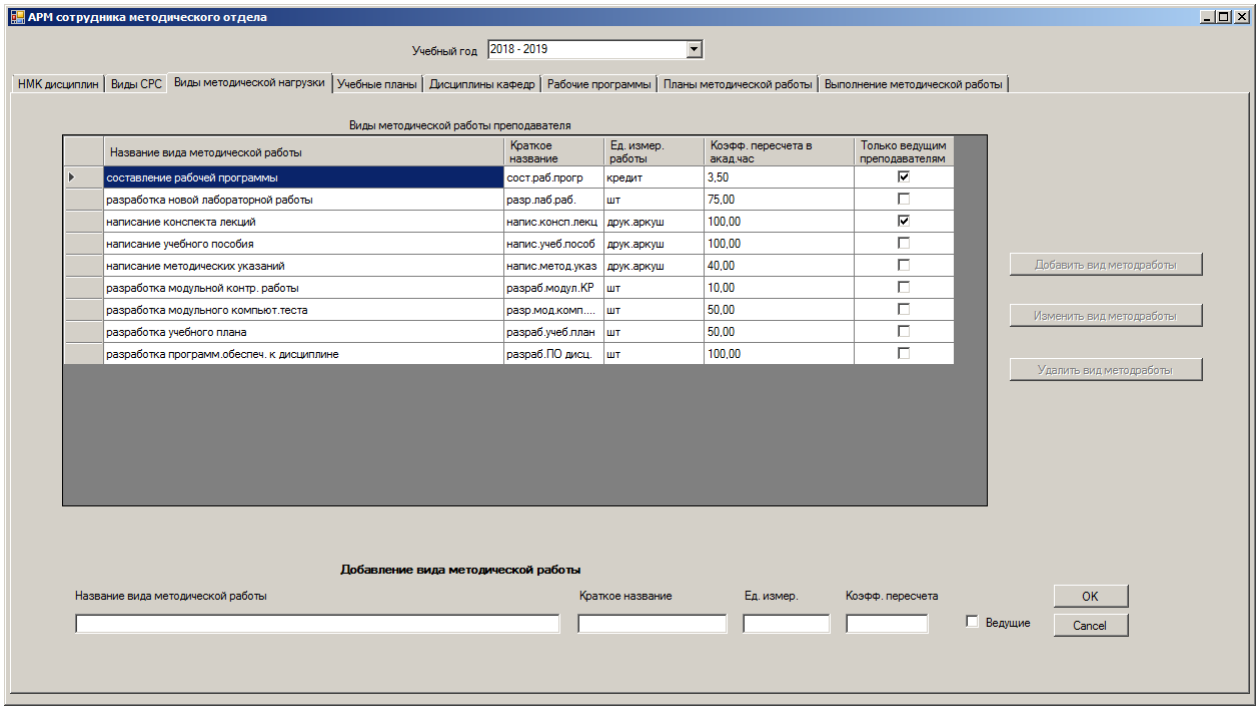

Рисунок 7.9 – Можливість додавання запису у таблицю видів методичної роботи викладачів

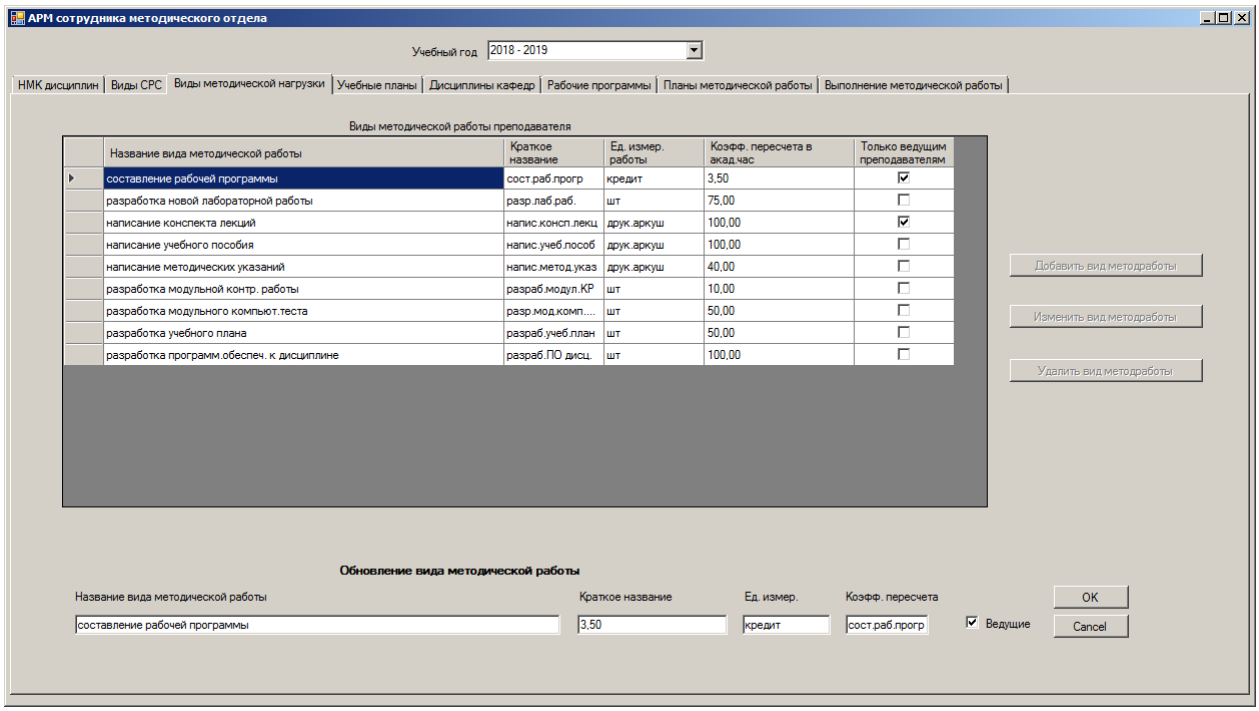

Рисунок 7.10 – Можливість оновлення запису у таблиці видів методичної роботи викладачів

На четвертій вкладці форми – «Учебные планы» – користувач повинен вибрати із випадного списку факультет, навчальні плани якого він хоче переглянути, (рис. 7.11).

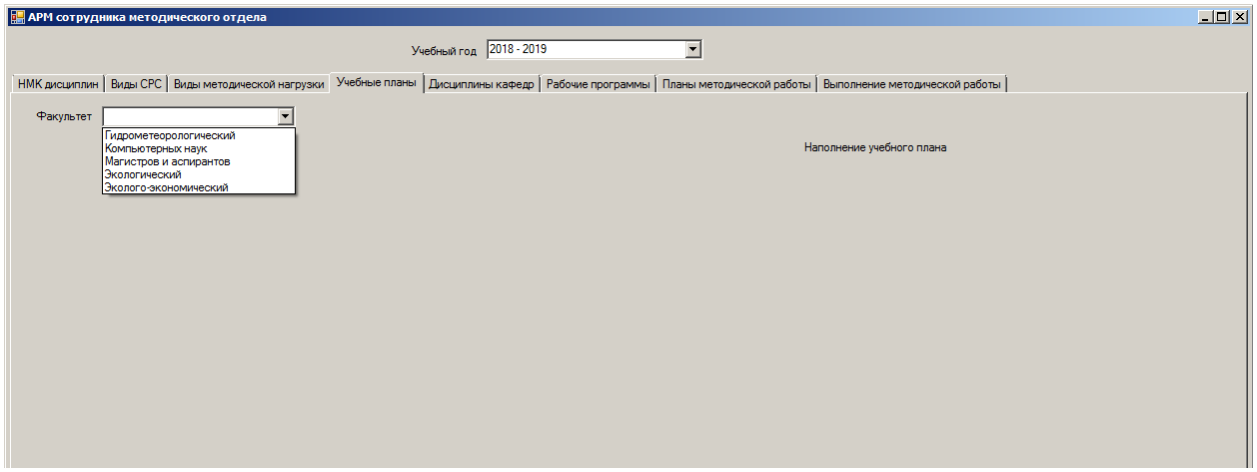

Рисунок 7.11 – Вибір факультету для перегляду навчальних планів

Після вибору факультету на формі стають видимими два табличних компонента: в лівій компонент виводиться список всіх навчальних планів вибраного факультету у вибраному навчальному році, в правий виводиться наповнення дисциплінами навчального плану з поточного рядка лівої таблиці (рис. 7.12).

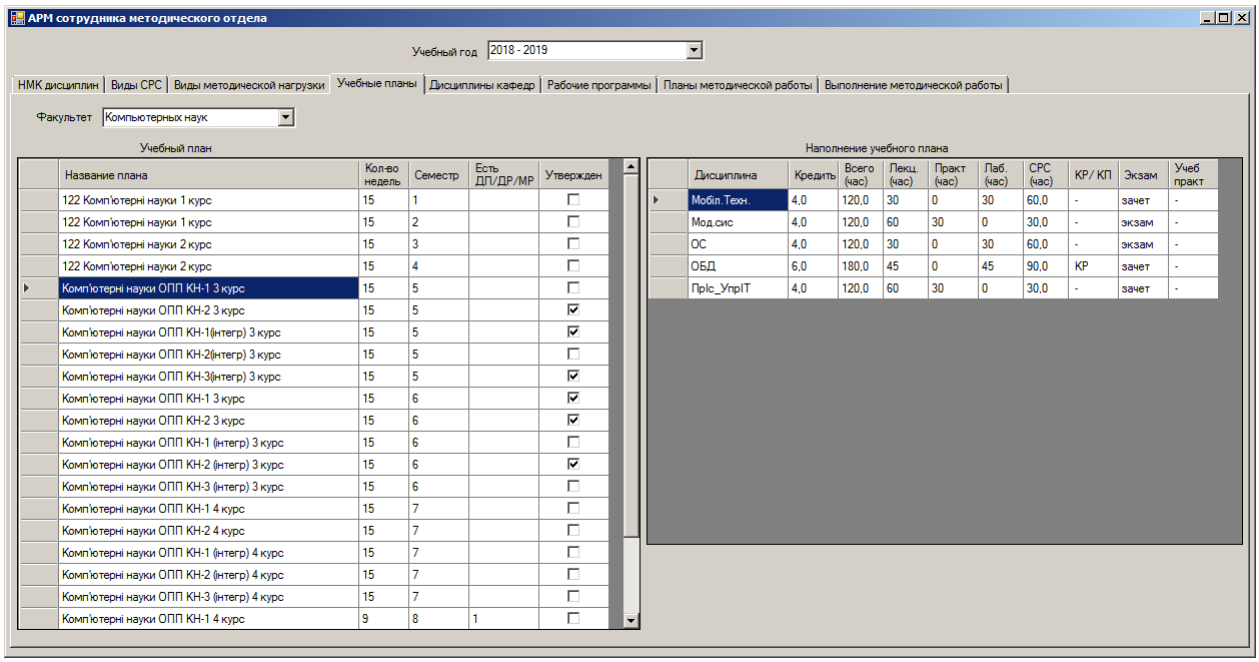

Рисунок 7.12 – Перегляд навчальних планів вибраного факультету

На п’ятій вкладці форми – «Дисциплины кафедр» – користувач повинен вибрати із випадного списку кафедру, дисципліни якої для вибраного навчального року він хоче переглянути, (рис. 7.13).

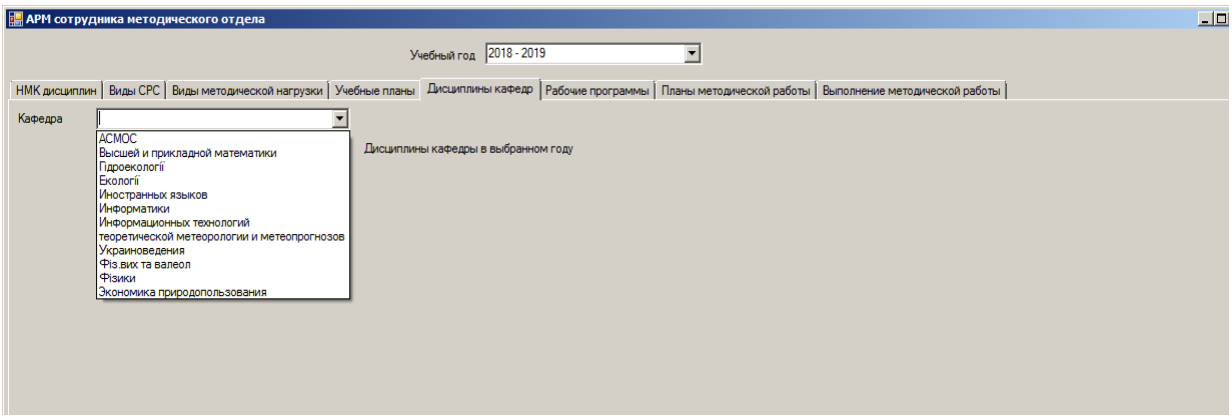

Рисунок 7.13 – Вибір кафедри для перегляду навчальних дисциплін

Список дисциплін кафедри включає в себе назву дисципліни, кількість кредитів, відведених під вивчення дисципліни, та посилання, восени або весною читається кожна дисципліна (рис. 7.14).

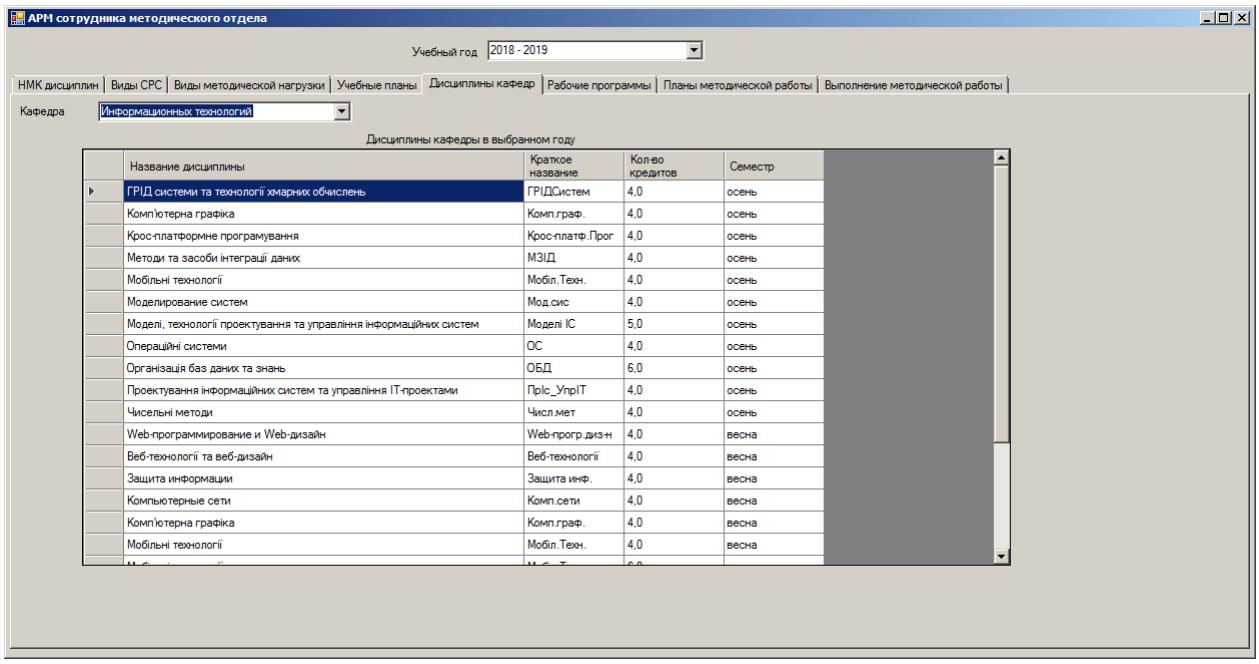

Рисунок 7.14 – Перегляд навчальних дисциплін вибраної кафедри

На шостій вкладці форми – «Рабочие программы» – користувач повинен вибрати із випадного списку кафедру, для дисципліни якої потрібно переглянути робочу програму (рис. 7.15).

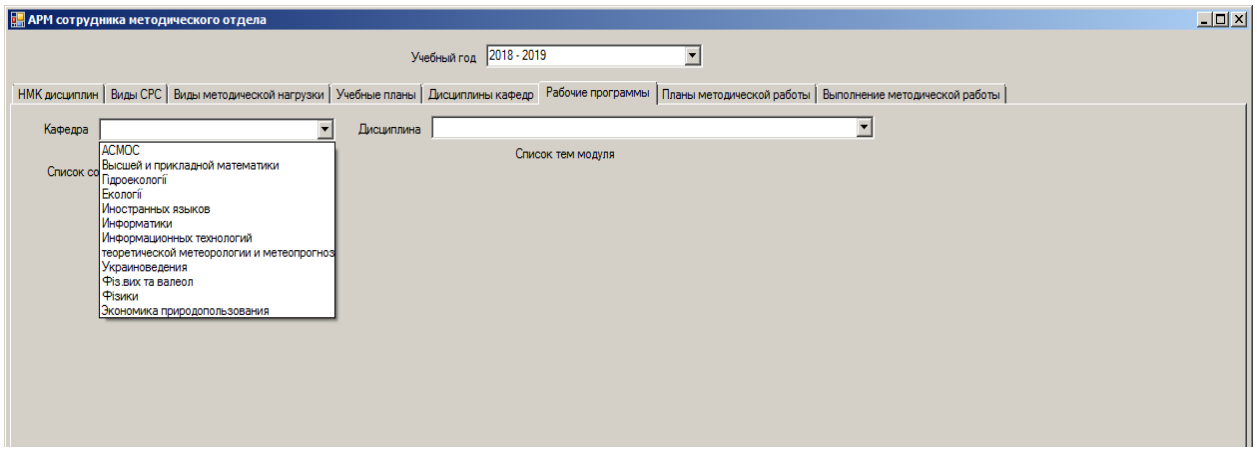

Рисунок 7.15 – Перегляд навчальних дисциплін вибраної кафедри

Потім для вибраної кафедри заповнюється випадний список з переліком дисциплін, закріплених за кафедрою у вибраному користувачем навчальному році (рис. 7.16).

Користувач повинен вибрати дисципліну, потім для цієї дисципліни на формі програми відображаються складові робочої програми: перелік модулів, перелік тем модулів та перелік робіт практичних модулів (рис. 7.17).

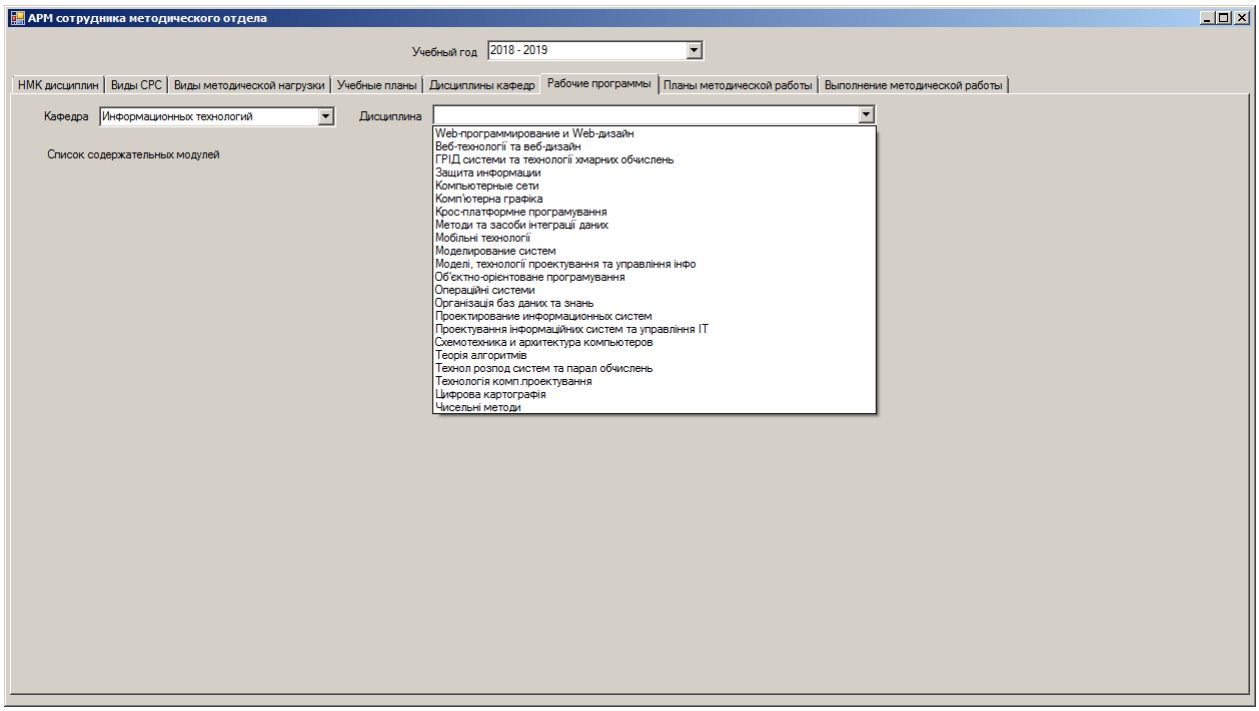

Рисунок 7.16 – Вибір користувачем дисципліни кафедри

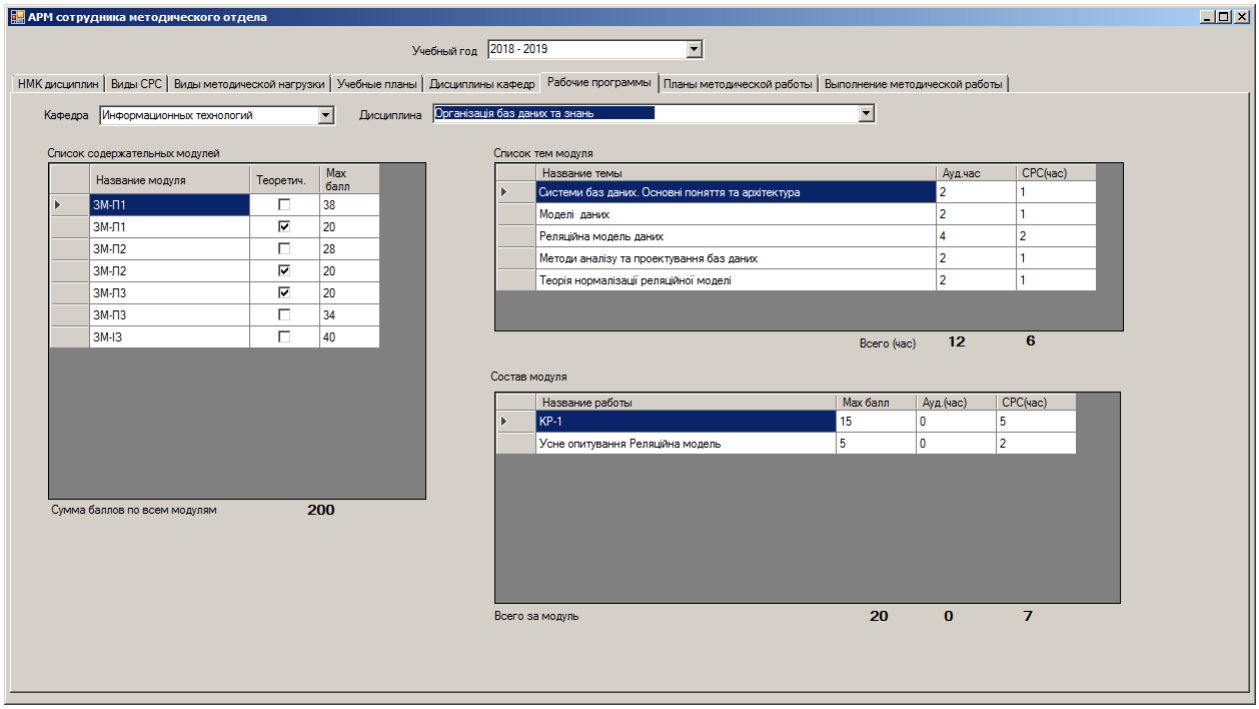

Рисунок 7.17 – Перегляд складових частин робочої програми вибраної навчальної дисципліни

Під правою таблицею зі списком змістовних модулів виводиться сумарна кількість балів, що може отримати студент при виконанні всіх контролюючих заходів всіх модулів дисципліни.

Під лівою таблицею з переліком тем поточного модулю виводиться сумарна кількість годин аудиторних занять та самостійної підготовки, відведених на вивчення всіх тем поточного лекційного модулю.

Під лівою таблицею з переліком робіт поточного модулю виводиться сумарна кількість годин аудиторних занять та самостійної підготовки, відведених на підготовку та виконання робіт (контролюючих заходів) модулю, а також сумарна кількість балів, що може отримати студент за виконання всіх контролюючих заходів даного модулю.

ВИСНОВКИ

Дана магістерська робота присвячена моделюванню та розробці підсистеми «Методичний відділ» інформаційної системи «Навчальний процес університету».

Розробка єдиної інформаційної системи, що описує навчальний процес в університеті від створення навчальних планів до проведення поточних та підсумкових контролюючих заходів дозволить скоротити потік документів між підрозділами університету, спростить складання необхідних документів, спростить перевірку планування навантаження викладачів та підведення підсумків виконання цього навантаження.

Методичний відділ університету вносить вагомий внесок у навчальний процес, забезпечує його методичну підтримку. Методичний відділ визначає пакети документів, необхідних для розробки робочих навчальних планів, проведення занять зі студентами, підтримки самостійної роботи студентів, проведення контролюючих заходів. Також методичний відділ визначає терміни оновлення пакетів методичних документів та контролює їх правильне оформлення та коректний зміст.

В екологічному державному університеті прийнята система перерахунку загального навантаження викладача: навчального, методичного, наукового, організаційного, виховної роботи, – в узагальнені академічні години. Методичним відділом університету розробляються таблиці перерахування кожного окремого виду загального навантаження у академічні години навантаження викладача.

Програма виду «Автоматизоване робоче місце працівника методичного відділу» допоможе у роботі працівникам цього відділу та спростить обмін документами між цим відділом, деканатами факультетів, кафедрами та іншими підрозділами університету.

Результатами виконання даної магістерської роботи є:

- розроблена концептуальна модель підсистеми «Методичний відділ» у складі ІС «Навчальний процес університету»;
- відповідно до моделі змінена схема бази даних база даних: додано 5 нових таблиці;
- розроблено нове серверне програмне забезпечення для підсистеми «Методичний відділ» до складу якого входять 9 збережених процедур та 6 уявлень на мові T-SQL для СУБД MS SQL Server 2008;

- розроблено клієнтське застосування «Автоматизоване робоче місце працівника методичного відділу» у вигляді офісного додатку до бази даних інформаційної системи «Навчальний процес університету»;
- застосування розроблене мовою C# у середовище Visual Studio 2015 у вигляді програми виду WindowsFormApplication;
- клієнтське застосування дозволяє працівнику методичного відділу переглядати та оновлювати дані у таблицях-довідниках методичної роботи та видів контролюючих заходів СРС;
- застосування дозволяє переглядати та затверджувати навчальні плани факультетів; переглядати робочі програми дисциплін, розроблені провідними викладачами; переглядати заплановане викладачами методичне навантаження та звіт про його виконання.

Програма «Автоматизоване робоче місце працівника методичного відділу» розроблялась як частина інформаційної системи «Навчальний процес університету», тому не вимагає введення великих масивів вихідних даних. Необхідні дані від підрозділів університету, повинні оновлюватись у базі даних інформаційної системи групами користувачів, що мають відповідні права на оновлення визначеного виду даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. Куликов Г. Г., Никулина Н. О., Речкалов А. В. Управление проектами на основе системного моделирования: Учебное пособие. Уфа: УГАТУ, 2009. – 171 с.
2. Вендров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем. М: «Финансы и статистика», 1998. – 98 с.
3. 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.
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. Сравнение SQL баз данных. http://webarty.net/databases/sravnenie-sql-baz-dannyh
6. Мамаев Е.В. Microsoft SQL Server 2000. – СПб.: БХВ-Петербург, 2004. – 1260 с.
7. Дэвидсон Л. Проектирование баз данных на SQL Server 2000 / Л.Дэвидсон; пер. с англ. – М.: БИНОМ. Лаборатория знаний, 2003. –680 с.
8. Роберт Виейра. Программирование баз данных Microsoft SQL Server 2005. Базовый курс. М.: «Диалектика», 2007. – с. 832.
9. Введение в MS SQL Server и T-SQL https://metanit.com/sql/sqlserver/1.1.php
10. Интегрированная среда разработки Visual Studio. URL: https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-studio-ide?view
11. Описание среды разработки Microsoft Visual Studio. URL: https://studbooks.net/2258619/informatika/opisanie_sredy_razrabotki_microsoft_visual_studio
12. Знакомство с Visual Studio. URL: https://habr.com/sandbox/79099.
13. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. RL: https://metanit.com/sharp/tutorial/
14. Руководство по программированию на C#. URL: https://docs.microsoft.com/ru-ru/previous-versions/67ef8sbd
15. Руководство по программированию на C#. URL: https://docs.microsoft.com/ru-ru/previous-versions/67ef8sbd

Д О Д А Т К И

ДОДАТОК А
ЛОГІЧНА СХЕМА БАЗИ ДАНИХ

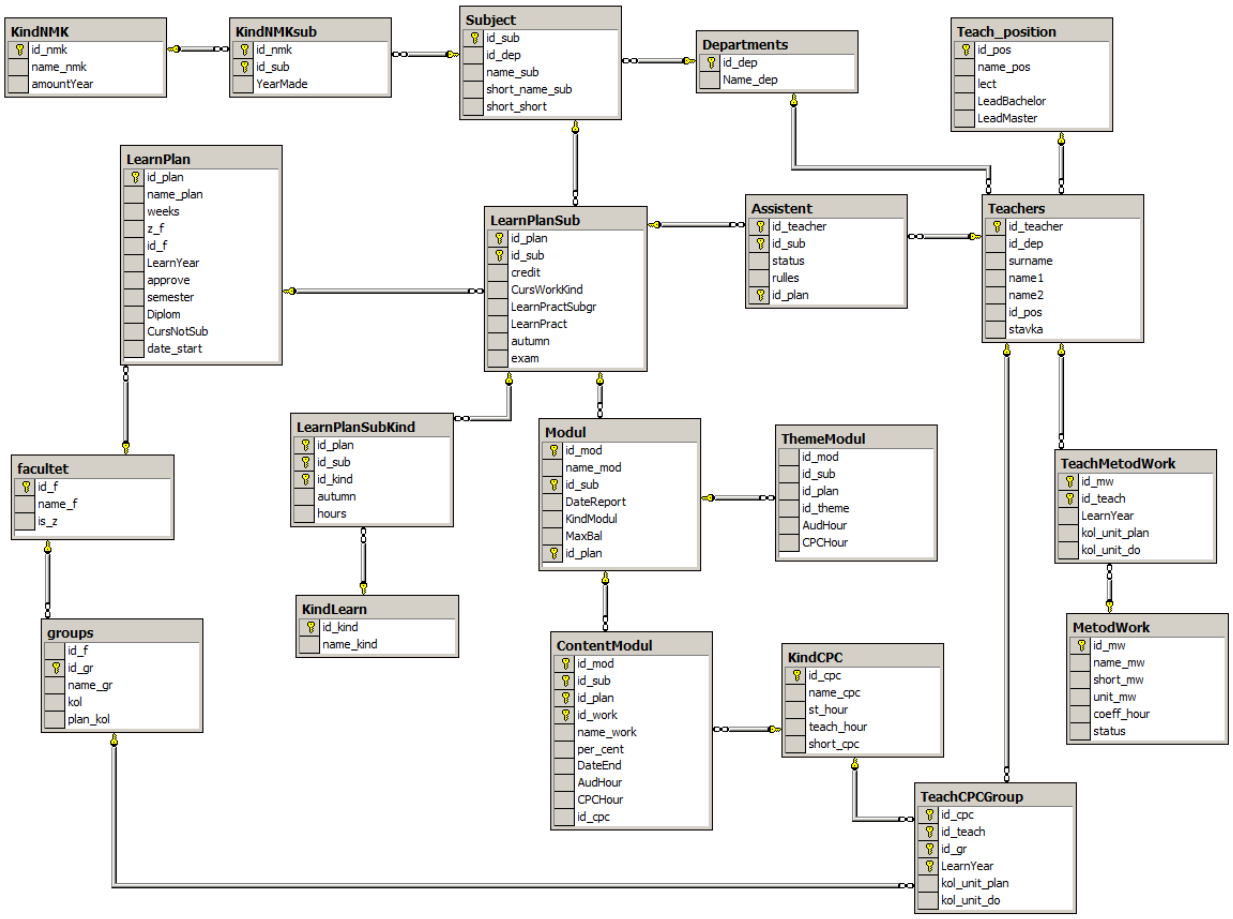

Рисунок А.1 – Підсхема «Методичний відділ» ІС
«Навчальний процес університету»

ДОДАТОК Б
ФІЗИЧНА СХЕМА БАЗИ ДАНИХ

```
create table KindNMK(
id_nmk smallint identity primary key,
name_nmk varchar (60) not null unique,
amountYear smallint
)
GO
CREATE  table  KindCPC (
id_cpc smallint identity primary key,
name_cpc varchar (60) not null unique ,
st_hour tinyint not null,
teach_hour decimal (4,2) not null
)
GO
Create table MetodWork(
id_mw smallint identity primary key,
name_mw varchar (60) not null unique ,
short_mw varchar (20) not null unique ,
unit_mw varchar (20) not null,
coeff_hour decimal (5,2) not null,
status bit
)
GO
CREATE  table  KindCPC (
id_cpc smallint identity primary key,
name_cpc varchar (60) not null unique ,
st_hour tinyint not null,
teach_hour decimal (4,2) not null
)
GO
Create table TeachCPCGroup(
id_cpc smallint references KindCPC ,
id_teach smallint not null references Teachers,
id_gr smallint not null references groups,
LearnYear smallint  not null,
kol_unit_plan tinyint  not null,
kol_unit_do tinyint  not null,
primary key ( id_cpc,id_teach, id_gr, LearnYear)
)
GO
```

ДОДАТОК В
СЕРВЕРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

```
CREATE procedure NMKAdd
  @id_nmk smallint output,
  @name_nmk varchar (60),
  @amountYear smallint,
  @k tinyint output,
  @msg varchar (50) output
as
begin
	set @k=(select k=count(*) from KindNMK
		where name_nmk = @name_nmk)
	if @k=0
	begin
		set @msg='OK'
		insert KindNMK (name_nmk, amountYear )
		values(@name_nmk,@amountYear )
		set @id_nmk = (select id_nmk from KindNMK
		where name_nmk = @name_nmk)
	end
	else
		set @msg='В БД есть метод.работа с названием  '+@name_nmk
end
GO

CREATE procedure NMKEdit
  @id_nmk smallint ,
  @name_nmk varchar (60),
  @amountYear smallint,
  @k tinyint output,
  @msg varchar (50) output
as
begin
	set @k=(select k=count(*) from KindNMK
		where name_nmk = @name_nmk and id_nmk != @id_nmk)
	if @k=0
	begin
		set @msg='OK'
		update KindNMK set  name_nmk = @name_nmk, amountYear = @amountYear
		where id_nmk = @id_nmk
	end
	else
		set @msg='В БД есть метод.работа с названием  '+@name_nmk
end
GO
CREATE  procedure NMKDel
  @id_nmk smallint ,
  @m tinyint,
  @k tinyint output,
  @msg varchar (50) output
as
begin
declare  @name_nmk varchar (60)
	if @m>0
	begin
```

```
        delete KindNMKsub where  id_nmk = @id_nmk
        delete KindNMK where @id_nmk = @id_nmk
        set @k=0
        set @msg = 'OK'
        end
        else
        begin
                set @k = (select k=count(*) from KindNMKsub where  id_nmk = @id_nmk)
                if @k = 0
                begin
                        set @msg = 'OK'
                        delete KindNMK where @id_nmk = @id_nmk
                end
                else
                begin
                        set @name_nmk= (select name_nmk from KindNMK where  id_nmk = @id_nmk)
                        set @msg='По дисциплинам разработано '+str(@k,3)+' работ этого вида.'+
                                char(13)+char(10)+'Все равно удалять вид НМК <'+@name_nmk+'>?'
                end
        end
end
GO
CREATE procedure CPCAdd
  @id_cpc smallint output,
  @name_cpc varchar (60),
  @st_hour tinyint ,
  @teach_hour decimal (4,2),
  @short_cpc varchar (20),
  @k tinyint output,
  @msg varchar (50) output
as
begin
        set @k=(select k=count(*) from KindCPC
                where name_cpc = @name_cpc)
        if @k=0
        begin
                set @msg='OK'
                insert KindCPC (name_cpc, st_hour, teach_hour, short_cpc )
                values(@name_cpc,@st_hour ,@teach_hour , @short_cpc)
                set @id_cpc = (select id_cpc from KindCPC
                where name_cpc = @name_cpc)
        end
        else
                set @msg='В БД есть вид СРС с названием '+@name_cpc
end
GO

CREATE procedure CPCEdit
  @id_cpc smallint ,
  @name_cpc varchar (60),
  @st_hour tinyint ,
  @teach_hour decimal (4,2),
  @short_cpc varchar (20),
  @k tinyint output,
  @msg varchar (50) output
as
begin
```

```
	set @k=(select k=count(*) from KindCPC
		where name_cpc = @name_cpc and id_cpc != @id_cpc)
	if @k=0
	begin
		set @msg='OK'
		update KindCPC set name_cpc = @name_cpc,st_hour = @st_hour,
teach_hour=@teach_hour, short_cpc=@short_cpc
		where id_cpc = @id_cpc
	end
	else
		set @msg='В БД есть вид СРС с названием '+@name_cpc
end
GO
CREATE  procedure CPCDel
  @id_cpc smallint ,
  @m tinyint,
  @k tinyint output,
  @msg varchar (50) output
as
begin
declare  @name_cpc varchar (60)
	if @m>0
	begin
	delete TeachCPCgroup where id_cpc = @id_cpc
	delete KindCPC where id_cpc = @id_cpc
	set @k=0
	set @msg = 'OK'
	end
	else
	begin
		set @k = (select k=count(*) from TeachCPCgroup where  id_cpc = @id_cpc)
		if @k = 0
		begin
			set @msg = 'OK'
			delete KindCPC where id_cpc = @id_cpc
		end
		else
		begin
			set @name_cpc= (select name_cpc from KindCPC where  id_cpc = @id_cpc)
			set @msg='У праподавателей запланировано '+str(@k,3)+' СРС этого вида.'+
				char(13)+char(10)+'Все равно удалять вид СРС <'+@name_cpc+'>?'
		end
	end
end
GO
```

ДОДАТОК Г
ВИХІДНИЙ КОД КЛІЄНТСЬКОГО ЗАСТОСУВАННЯ

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Metodical
{
    public partial class Form1 : Form
    {
        short LearnYear = 2017, id_teach = 66, id_dep = 1, id_plan = 69, id_gr = 67,
id_sub = 65, id_f = 1;
        string name = "";
        int aut = 1;
        bool a_cpc = true, a_nmk = true, a_mw = true;
        bool prevE = false, prevD = false, autumn = true;
        private System.Data.OleDb.OleDbCommand addNMKCmd;
        private System.Data.OleDb.OleDbCommand editNMKCmd;
        private System.Data.OleDb.OleDbCommand delNMKCmd;
        private System.Data.OleDb.OleDbCommand addCPCCmd;
        private System.Data.OleDb.OleDbCommand editCPCCmd;
        private System.Data.OleDb.OleDbCommand delCPCCmd;
        private System.Data.OleDb.OleDbCommand addMWCmd;
        private System.Data.OleDb.OleDbCommand editMWCmd;
        private System.Data.OleDb.OleDbCommand delMWCmd;
        System.Data.OleDb.OleDbConnection Connection;

        public Form1()
        {
            InitializeComponent();
            Connection = this.kindNMKTableAdapter.Connection;
            fillDepComboBox();
            fillFacComboBox();
            init_addMWCmd(); init_editMWCmd();init_delMWCmd();
            init_addCPCCmd(); init_editCPCCmd();init_delCPCCmd();
            init_addNMKCmd(); init_editNMKCmd();init_delNMKCmd();
            this.learnPlanTableAdapter.Fill(this.dataSet1.LearnPlan, id_f, LearnYear);
            this.lP_viewTableAdapter.Fill(this.dataSet1.LP_view, id_plan);
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            this.metodWorkTableAdapter.Fill(this.dataSet1.MetodWork);
            this.kindCPCTableAdapter.Fill(this.dataSet1.KindCPC);
            this.kindNMKTableAdapter.Fill(this.dataSet1.KindNMK);
            this.lP_viewTableAdapter.Fill(this.dataSet1.LP_view, id_plan);
            this.learnPlanTableAdapter.Fill(this.dataSet1.LearnPlan, id_f, LearnYear);
            this.dataTable1TableAdapter.Fill(this.dataSet1.DataTable1, LearnYear,
id_dep);
            this.modulTableAdapter.Fill(this.dataSet1.Modul, id_sub);
        }

        private void init_addNMKCmd()
        {
            addNMKCmd = new System.Data.OleDb.OleDbCommand();
            addNMKCmd.Connection = Connection;
            addNMKCmd.CommandText = "NMKAdd";
```

```
            addNMKCmd.CommandType = System.Data.CommandType.StoredProcedure;
            addNMKCmd.Parameters.Add("@id_nmk", Sys-
tem.Data.OleDb.OleDbType.SmallInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            addNMKCmd.Parameters.Add("@name_nmk", System.Data.OleDb.OleDbType.VarChar,
60);
            addNMKCmd.Parameters.Add("@amountYear", Sys-
tem.Data.OleDb.OleDbType.SmallInt);
            addNMKCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            addNMKCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }

        private void init_editNMKCmd()
        {
            editNMKCmd = new System.Data.OleDb.OleDbCommand();
            editNMKCmd.Connection = Connection;
            editNMKCmd.CommandText = "NMKEdit";
            editNMKCmd.CommandType = System.Data.CommandType.StoredProcedure;
            editNMKCmd.Parameters.Add("@id_nmk", System.Data.OleDb.OleDbType.SmallInt);
            editNMKCmd.Parameters.Add("@name_nmk", System.Data.OleDb.OleDbType.VarChar,
60);
            editNMKCmd.Parameters.Add("@amountYear", Sys-
tem.Data.OleDb.OleDbType.SmallInt);
            editNMKCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            editNMKCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }
        private void init_delNMKCmd()
        {
            delNMKCmd = new System.Data.OleDb.OleDbCommand();
            delNMKCmd.Connection = Connection;
            delNMKCmd.CommandText = "NMKDel";
            delNMKCmd.CommandType = System.Data.CommandType.StoredProcedure;
            delNMKCmd.Parameters.Add("@id_nmk", System.Data.OleDb.OleDbType.SmallInt);
            delNMKCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            delNMKCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }

        private void init_addCPCCmd()
        {
            addCPCCmd = new System.Data.OleDb.OleDbCommand();
            addCPCCmd.Connection = Connection;
            addCPCCmd.CommandText = "CPCAdd";
            addCPCCmd.CommandType = System.Data.CommandType.StoredProcedure;
            addCPCCmd.Parameters.Add("@id_cpc", System.Data.OleDb.OleDbType.SmallInt);
            addCPCCmd.Parameters.Add("@name_cpc", System.Data.OleDb.OleDbType.VarChar,
60);
            addCPCCmd.Parameters.Add("@st_hour", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt);
            addCPCCmd.Parameters.Add("@teach_hour", System.Data.OleDb.OleDbType.Decimal);
            addCPCCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            addCPCCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }
```

```
        private void init_editCPCCmd()
        {
            editCPCCmd = new System.Data.OleDb.OleDbCommand();
            editCPCCmd.Connection = Connection;
            editCPCCmd.CommandText = "CPCEdit";
            editCPCCmd.CommandType = System.Data.CommandType.StoredProcedure;
            editCPCCmd.Parameters.Add("@id_cpc", System.Data.OleDb.OleDbType.SmallInt);
            editCPCCmd.Parameters.Add("@name_cpc", System.Data.OleDb.OleDbType.VarChar,
60);
            editCPCCmd.Parameters.Add("@st_hour", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt);
            editCPCCmd.Parameters.Add("@teach_hour", Sys-
tem.Data.OleDb.OleDbType.Decimal);
            editCPCCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            editCPCCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }
        private void init_delCPCCmd()
        {
            delCPCCmd = new System.Data.OleDb.OleDbCommand();
            delCPCCmd.Connection = Connection;
            delCPCCmd.CommandText = "CPCDel";
            delCPCCmd.CommandType = System.Data.CommandType.StoredProcedure;
            delCPCCmd.Parameters.Add("@id_cpc", System.Data.OleDb.OleDbType.SmallInt);
            delCPCCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            delCPCCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }

        private void init_addMWCmd()
        {
            addMWCmd = new System.Data.OleDb.OleDbCommand();
            addMWCmd.Connection = Connection;
            addMWCmd.CommandText = "MetodWAdd";
            addMWCmd.CommandType = System.Data.CommandType.StoredProcedure;
            addMWCmd.Parameters.Add("@id_mw", Sys-
tem.Data.OleDb.OleDbType.SmallInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            addMWCmd.Parameters.Add("@name_mw", System.Data.OleDb.OleDbType.VarChar, 60);
            addMWCmd.Parameters.Add("@short_mw", System.Data.OleDb.OleDbType.VarChar,
20);
            addMWCmd.Parameters.Add("@unit_mw", System.Data.OleDb.OleDbType.VarChar, 20);
            addMWCmd.Parameters.Add("@coeff_hour", System.Data.OleDb.OleDbType.Decimal);
            addMWCmd.Parameters.Add("@status", System.Data.OleDb.OleDbType.Boolean);
            addMWCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            addMWCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }

        private void init_editCPCCmd()
        {
            editMWCmd = new System.Data.OleDb.OleDbCommand();
            editMWCmd.Connection = Connection;
            editMWCmd.CommandText = "MetodWEdit";
            editMWCmd.CommandType = System.Data.CommandType.StoredProcedure;
            editMWCmd.Parameters.Add("@id_mw", System.Data.OleDb.OleDbType.SmallInt);
            editMWCmd.Parameters.Add("@name_mw", System.Data.OleDb.OleDbType.VarChar,
60);
```

```
            editMWCmd.Parameters.Add("@short_mw", System.Data.OleDb.OleDbType.VarChar,
20);
            editMWCmd.Parameters.Add("@unit_mw", System.Data.OleDb.OleDbType.VarChar,
20);
            editMWCmd.Parameters.Add("@coeff_hour", System.Data.OleDb.OleDbType.Decimal);
            editMWCmd.Parameters.Add("@status", System.Data.OleDb.OleDbType.Boolean);
            editMWCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            editMWCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }
        private void init_delCPCCmd()
        {
            delMWCmd = new System.Data.OleDb.OleDbCommand();
            delMWCmd.Connection = Connection;
            delMWCmd.CommandText = "MetodWDel";
            delMWCmd.CommandType = System.Data.CommandType.StoredProcedure;
            delMWCmd.Parameters.Add("@id_mw", System.Data.OleDb.OleDbType.SmallInt);
            delMWCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            delMWCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }

        private void fillDepComboBox()
        {
            Connection = this.metodWorkTableAdapter.Connection;
            System.Data.OleDb.OleDbCommand depCmd = new Sys-
tem.Data.OleDb.OleDbCommand("Select id_dep, name_dep from Departments", Connection);
            System.Data.ConnectionState previousConnectionState = Connection.State;
            if (((Connection.State & System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open))
            {
                Connection.Open();

            }
            try
            {
                System.Data.OleDb.OleDbDataReader reader = depCmd.ExecuteReader();
                List_Item li;
                depComboBox.Items.Clear(); depComboBox2.Items.Clear();
                depComboBox3.Items.Clear();depComboBox4.Items.Clear();
                int i = 0;
                while (reader.Read())
                {
                    li = new List_Item(reader.GetInt16(0), reader.GetString(1));
                    depComboBox.Items.Add(li); depComboBox2.Items.Add(li);
                    depComboBox3.Items.Add(li); depComboBox4.Items.Add(li);
                    ++i;
                }
                reader.Close();
                depComboBox.SelectedIndex = -1; depComboBox2.SelectedIndex = -1;
                depComboBox3.SelectedIndex = -1;depComboBox4.SelectedIndex = -1;

            }
            finally
            {
                if ((previousConnectionState == System.Data.ConnectionState.Closed))
                    Connection.Close();
            }
        }

        private void fillFacComboBox()
```

```
        {
            Connection = this.metodWorkTableAdapter.Connection;
            System.Data.OleDb.OleDbCommand facCmd = new Sys-
tem.Data.OleDb.OleDbCommand("Select id_f, name_f from facultet where is_z=0 order by
name_f", Connection);
            System.Data.ConnectionState previousConnectionState = Connection.State;
            if (((Connection.State & System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open))
            {
                Connection.Open();
            }
            try
            {
                System.Data.OleDb.OleDbDataReader reader = facCmd.ExecuteReader();
                List_Item li;
                facComboBox.Items.Clear();
                int i = 0;
                while (reader.Read())
                {
                    li = new List_Item(reader.GetInt16(0), reader.GetString(1));
                    facComboBox.Items.Add(li);
                    ++i;
                }
                reader.Close();
                facComboBox.SelectedIndex = -1;
            }
            finally
            {
                if ((previousConnectionState == System.Data.ConnectionState.Closed))
                    Connection.Close();
            }
        }

        private void fillTeachComboBox(short id_dep)
        {
            Connection = this.metodWorkTableAdapter.Connection;
            System.Data.OleDb.OleDbCommand teachCmd = new Sys-
tem.Data.OleDb.OleDbCommand("Select id_teacher, prep=surname+' '+name1+' '+name2 from
Teachers where id_dep=" + id_dep.ToString() + " order by prep", Connection);
            System.Data.ConnectionState previousConnectionState = Connection.State;
            if (((Connection.State & System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open))
            {
                Connection.Open();

            }
            try
            {
                System.Data.OleDb.OleDbDataReader reader = teachCmd.ExecuteReader();
                List_Item li;
                teachComboBox1.Items.Clear();
                int i = 0;
                while (reader.Read())
                {
                    li = new List_Item(reader.GetInt16(0), reader.GetString(1));
                    teachComboBox1.Items.Add(li);
                    ++i;
                }
                reader.Close();
                teachComboBox1.SelectedIndex = -1;

            }
            finally
            {
                if ((previousConnectionState == System.Data.ConnectionState.Closed))
```

```
                Connection.Close();
            }
        }

        private void fillSubComboBox(short id_dep)
        {
            Connection = this.metodWorkTableAdapter.Connection;
            System.Data.OleDb.OleDbCommand subCmd = new Sys-
tem.Data.OleDb.OleDbCommand("Select id_sub, name_s=convert(char(50), name_sub) from Sub-
ject where id_dep="

+ id_dep.ToString() + " order by name_s", Connection);
            System.Data.ConnectionState previousConnectionState = Connection.State;
            if (((Connection.State & System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open))
            {
                Connection.Open();

            }
            try
            {
                System.Data.OleDb.OleDbDataReader reader = subCmd.ExecuteReader();
                List_Item li;
                subComboBox.Items.Clear();
                int i = 0;
                while (reader.Read())
                {
                    li = new List_Item(reader.GetInt16(0), reader.GetString(1));
                    subComboBox.Items.Add(li);
                    ++i;
                }
                reader.Close();
                subComboBox.SelectedIndex = -1;

            }
            finally
            {
                if ((previousConnectionState == System.Data.ConnectionState.Closed))
                    Connection.Close();
            }
        }

        private void comboBox4_SelectedIndexChanged(object sender, EventArgs e)
        {
            string ms = comboBox4.SelectedItem.ToString();
            string[] d = new string[3];
            d = ms.Split(' ');

            LearnYear = Convert.ToInt16(d[0]);
            if (LearnYear > 2017) LearnYear = 2017;
            tabControl1.Visible = true;
        }

        private void dataGridView1_CellEnter(object sender, DataGridViewCellEventArgs e)
        {
            id_plan = (short)(dataGridView1.Rows[e.RowIndex].Cells[7].Value);
            this.lP_viewTableAdapter.Fill(this.dataSet1.LP_view, id_plan);
        }

        private void depComboBox3_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_dep = (short)(((List_Item)(depComboBox3.SelectedItem)).id);
            fillTeachComboBox(id_dep);
        }
```

```
        private void depComboBox4_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_dep = (short)(((List_Item)(depComboBox4.SelectedItem)).id);
            fillTeachComboBox(id_dep);
        }

        private void facComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_f = (short)(((List_Item)(facComboBox.SelectedItem)).id);
            this.learnPlanTableAdapter.Fill(this.dataSet1.LearnPlan, id_f, LearnYear);
            if (dataGridView1.RowCount > 0)
            {
                id_plan = (short)(dataGridView1.Rows[0].Cells[7].Value);
                this.lP_viewTableAdapter.Fill(this.dataSet1.LP_view, id_plan);
                dataGridView2.Visible = true;
            }
            dataGridView1.Visible = true;
        }

        private void depComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_dep = (short)(((List_Item)(depComboBox.SelectedItem)).id);
            this.dataTable1TableAdapter.Fill(this.dataSet1.DataTable1, LearnYear,
id_dep);
            dataGridView6.Visible = true;
        }

        private void depComboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_dep = (short)(((List_Item)(depComboBox2.SelectedItem)).id);
            fillSubComboBox(id_dep);
        }

        private void subComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_sub = (short)(((List_Item)(subComboBox.SelectedItem)).id);
            this.modulTableAdapter.Fill(this.dataSet1.Modul, id_sub); data-
GridView7.Visible = true;
        }

        private void editNMKButton_Click(object sender, EventArgs e)
        {
            string s = (string)(dataGridView3.CurrentRow.Cells[0].Value);
            short y = (short)(dataGridView3.CurrentRow.Cells[1].Value);
            nameNMKTextBox.Text = s;
            amountTextBox.Text = y.ToString();
            p1(true);
            a_nmk = false;
        }

        private void p1(bool vis)
        {
            delNMKButton.Enabled = !vis;
            editNMKButton.Enabled = !vis;
            addNMKButton.Enabled = !vis;
            label18.Visible = vis;nameNMKLabel.Visible = vis;
            amountLabel.Visible = vis;nameNMKTextBox.Visible = vis;
            amountTextBox.Visible = vis;
            OKbutton1.Visible = vis;CancelButton1.Visible = vis;
        }



        private void addCPCButton_Click(object sender, EventArgs e)
        {
```

```
            delCPCButton.Enabled = false;  addCPCButton.Enabled = false;
            editCPCButton.Enabled = false;
        }

        private void editCPCButton_Click(object sender, EventArgs e)
        {
            string s = (string)(dataGridView4.CurrentRow.Cells[0].Value);
            string ss = (string)(dataGridView4.CurrentRow.Cells[1].Value);
            byte y = (byte)(dataGridView4.CurrentRow.Cells[2].Value);
            decimal h = (decimal)(dataGridView4.CurrentRow.Cells[3].Value);
            textBox1.Text = s;textBox2.Text = ss;
            textBox3.Text = y.ToString();textBox4.Text = h.ToString();
            delCPCButton.Enabled = false;  addCPCButton.Enabled = false;
            editCPCButton.Enabled = false;
        }

        private void button11_Click(object sender, EventArgs e)
        {
            string s = (string)(dataGridView5.CurrentRow.Cells[0].Value);
            string ss = (string)(dataGridView5.CurrentRow.Cells[1].Value);
            string y = (string)(dataGridView5.CurrentRow.Cells[2].Value);
            decimal h = (decimal)(dataGridView5.CurrentRow.Cells[3].Value);
            bool c= (bool)(dataGridView5.CurrentRow.Cells[4].Value);
            textBox5.Text = s;textBox6.Text = ss;
            textBox7.Text = y;textBox8.Text = h.ToString();
            checkBox1.Checked = c;
            button11.Enabled = false;button10.Enabled = false;button12.Enabled = false;
        }

        private void button12_Click(object sender, EventArgs e)
        {
            textBox5.Text = "";textBox6.Text = "";
            textBox7.Text = "";textBox8.Text = "";
            checkBox1.Checked = false;
            button11.Enabled = false;button10.Enabled = false;button12.Enabled = false;

        }

        private void dataGridView7_CellEnter(object sender, DataGridViewCellEventArgs e)
        {

        }

        private void dataGridView7_RowEnter(object sender, DataGridViewCellEventArgs e)
        {
            id_mod = (short)(dataGridView7.Rows[e.RowIndex].Cells[3].Value);
            id_plan = (short)(dataGridView7.Rows[e.RowIndex].Cells[5].Value);
            this.nameThemeModulTableAdapter.Fill(this.dataSet11.NameThemeModul, id_sub,
id_plan, id_mod);
            dataGridView7.Visible = true;
            this.sumBalModulTableAdapter.Fill(this.dataSet11.SumBalModul, id_sub);
            this.dataTable5TableAdapter.Fill(dataSet11.DataTable5, id_sub, id_plan,
id_mod);
            this.dataTable6TableAdapter.Fill(dataSet11.DataTable6, id_sub, id_plan,
id_mod);
            this.contentModulTableAdapter.Fill(dataSet11.ContentModul, id_sub, id_plan,
id_mod);
            int m = dataGridView9.RowCount;
            if (m > 0)
            {
                balModlabel.Text =
((int)(dataGridView9.Rows[0].Cells[0].Value)).ToString();
                audHModlabel.Text =
((int)(dataGridView9.Rows[0].Cells[1].Value)).ToString();
```

```
                CPCHModlabel.Text =
((int)(dataGridView9.Rows[0].Cells[2].Value)).ToString();
            }
            else
            {
                balModlabel.Text = " ";
                audHModlabel.Text = " ";
                CPCHModlabel.Text = " ";
            }


            int n = dataGridView8.RowCount;
            if (n > 0)
            {
                audHlabel.Text =
((int)(dataGridView8.Rows[0].Cells[0].Value)).ToString();
                CPCHlabel.Text =
((int)(dataGridView8.Rows[0].Cells[1].Value)).ToString();
                balLabel.Text = ((int)(dataGridView6.Rows[0].Cells[0].Value)).ToString();
            }
            else
            {
                audHlabel.Text = "0"; CPCHlabel.Text = "0";
            }
        }

        private void addNMKButton_Click(object sender, EventArgs e)
        {
            a_nmk = true;p1(true);
        }

        private void CancelButton1_Click(object sender, EventArgs e)
        {
            p1(false);
        }

        private void OKbutton1_Click(object sender, EventArgs e)
        {
            short amount = Convert.ToInt16(amountTextBox.Text);
            short id_nmk = (short)(dataGridView3.CurrentRow.Cells[2].Value);
            string name_nmk = nameNMKTextBox.Text;
            if (a == 0)
            {
                addNMKCmd.Parameters["@id_theme"].Value = amount;
                addNMKCmd.Parameters["@id_nmk"].Value = id_nmk;
                addNMKCmd.Parameters["@name_nmk"].Value = name_nmk;
                addNMKCmd.Parameters["@k"].Value = System.DBNull.Value;
                addNMKCmd.Parameters["@msg"].Value = System.DBNull.Value;
                System.Data.ConnectionState previousConnectionState =
addNMKCmd.Connection.State;
                if (((addNMKCmd.Connection.State & System.Data.ConnectionState.Open)
                            != System.Data.ConnectionState.Open))
                {
                    addNMKCmd.Connection.Open();
                }
                try
                {
                    addNMKCmd.ExecuteNonQuery();
                    if ((byte)(addNMKCmd.Parameters["@k"].Value) > 0)
                        MessageBox.Show((string)(addNMKCmd.Parameters["@msg"].Value),
"Ошибка добавления темы", MessageBoxButtons.OK);
                    else
                        this.kindNMKTableAdapter.Fill(this.dataSet1.KindNMK);
                }
                finally
```

```
                {
                    if ((previousConnectionState == System.Data.ConnectionState.Closed))
                        addNMKCmd.Connection.Close();
                }
            }
            else
            {
                editNMKCmd.Parameters["@id_nmk"].Value = id_nmk;
                editNMKCmd.Parameters["@amountYear"].Value = amount;
                editNMKCmd.Parameters["@name_nmk"].Value = name_nmk;
                editNMKCmd.Parameters["@k"].Value = System.DBNull.Value;
                editNMKCmd.Parameters["@msg"].Value = System.DBNull.Value;
                System.Data.ConnectionState previousConnectionState = edit-
NMKCmd.Connection.State;
                if (((editNMKCmd.Connection.State & System.Data.ConnectionState.Open)
                            != System.Data.ConnectionState.Open))
                {
                    editNMKCmd.Connection.Open();
                }
                try
                {
                    editNMKCmd.ExecuteNonQuery();
                    if ((byte)(editNMKCmd.Parameters["@k"].Value) > 0)
                        MessageBox.Show((string)(editNMKCmd.Parameters["@msg"].Value),
"Ошибка редактирования темы", MessageBoxButtons.OK);
                    else
                        this.kindNMKTableAdapter.Fill(this.dataSet1.KindNMK);
                }
                finally
                {
                    if ((previousConnectionState == System.Data.ConnectionState.Closed))
                        editNMKCmd.Connection.Close();
                }
            }

            p1(false);
        }


    }
    public class List_Item : Object
    {
        public int id;
        public string name;
        public List_Item(int id, string name)
        {
            this.id = id;
            this.name = name;
        }
        public override string ToString()
        {
            return name;
        }
    }
}
```