МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет магістерської та
аспірантської підготовки
Кафедра інформаційних
технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Інтеграція СБД «Електронний деканат та тестуюча система»

Виконав студент 2 курсу групи МК-61
спеціальності 8.05010101 Інформаційні
управляючі системи та технології,
Біличенко Мирослав Олександрович

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

Консультант

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

Одеса 2017

АНОТАЦІЯ

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

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

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

Об’єктом дослідження є навчальний процес вищого навчального закладу.

Предметом дослідження є концептуальне моделювання навчального процесу вищого навчального закладу.

Методи дослідження. Використовуються методи теорії відношень та реляційної алгебри.

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

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

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

Практичним значенням є розробка бази даних навчального процесу ВНЗ.

Результати даної роботи можуть використовуватись у вищих навчальних закладах.

Вихідні дані: інформація про навчальний процес ВНЗ.

Об’єм роботи складає – 77 стор., кількість рисунків – 24, кількість використаної літератури – 15.

Ключові слова: база даних, навчальний процес, тестуюча система.

SUMMARY

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

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

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

The object is the educational process of higher educational institutions.

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

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

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

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

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

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

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

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

The volume of work is – 77 p., the number of figures – 24, the number of references – 15.

Keywords: database, the learning process, testing system.

ЗМІСТ

ВСТУП 9
1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ 10
2 СИСТЕМНИЙ АНАЛІЗ І КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 13
2.1 Етапи проектування бази даних 13
2.2 Концептуальне проектування бази даних 13
2.2.1 Визначення основних типів сутностей 13
2.2.2 Визначення атрибутів і зв’язування їх з типами сутностей і зв’язків 22
2.2.3 Визначення відповідності концептуальної моделі транзакціям користувачів 23
3 ВИБІР СУБД І СЕРЕДОВИЩА РОЗРОБКИ КЛІЄНТСЬКОГО ДОДАТКУ 26
3.1 Вибір СУБД 26
3.2 Вибір середовища розробки кліентського додадтку до БД 31
4 ЛОГІЧНЕ І ФІЗИЧНЕ ПРОЕКТУВАННЯ БД 38
4.1 Логічне проектування 38
4.2 Фізичне проектування. 40
5 СЕРВЕРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ 43
6 ОПИС РОЗРОБЛЕНОГО ДОДАТКУ ДИСПЕТЧЕРУ 49
6.1 Загальні відомості 49
6.2 Функціональні призначення 49
6.3 Опис програми 49
6.3.1 Керівництво програміста 49
6.3.2 Керівництво користувача 52
ВИСНОВКИ 63
ПЕРЕЛІК ПОСИЛАНЬ 64
Д О Д А Т К И 65
ДОДАТОК Б – ЛОГІЧНА СХЕМА БД: ПІДСХЕМА «ЕЛЕКТРОННИЙ ДЕКАНАТ ТА ПОТОЧНА УСПІШНІСТЬ» 67
ДОДАТОК В – ФІЗИЧНА СХЕМА БД: ТАБЛИЦІ ТА ПРЕДСТАВЛЕННЯ ДЛЯ ПІДСХЕМИ «ЕЛЕКТРОННИЙ ДЕКАНАТ ТА ПОТОЧНА УСПІШНІСТЬ» 68
ДОДАТОК Г – СЕРВЕРНЕ ПЗ: ЗБЕРЕЖЕНІ ПРОЦЕДУРИ ДЛЯ ПІДСХЕМИ «ЕЛЕКТРОННИЙ ДЕКАНАТ ТА ПОТОЧНА УСПІШНІСТЬ» 69
ДОДАТОК Г – ВИХІДНИЙ КОД КЛІЄНТСЬКОГО ЗАСТОСУВАННЯ 72

ВСТУП

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

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

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

Задачею даної частини комплексної магістерської роботи є інтегрування у єдину базу даних інформаційної системи «Університет» баз даних «Електронний деканат», «Тестуюча система» та «Інтегральні відомості», а також розробка офісного клієнтського застосування для групи користувачів «Працівник деканату».

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Для представлення інформації про університет різних групам користувачів необхідно створити інформаційну систему (ІС) «Університет». ІС «Університет» повинна опиратися на базу даних (або набір баз даних) та мати у своєму складі набір веб-застосувань, що будуть входити у веб-портал університету. Також у складі ІС буде набір офісних застосувань, які будуть використовуватись як для перегляду інформації в базі даних працівниками університету, так і для оновлення цієї інформації особами, які будуть мати відповідні права (рис. 1.1).

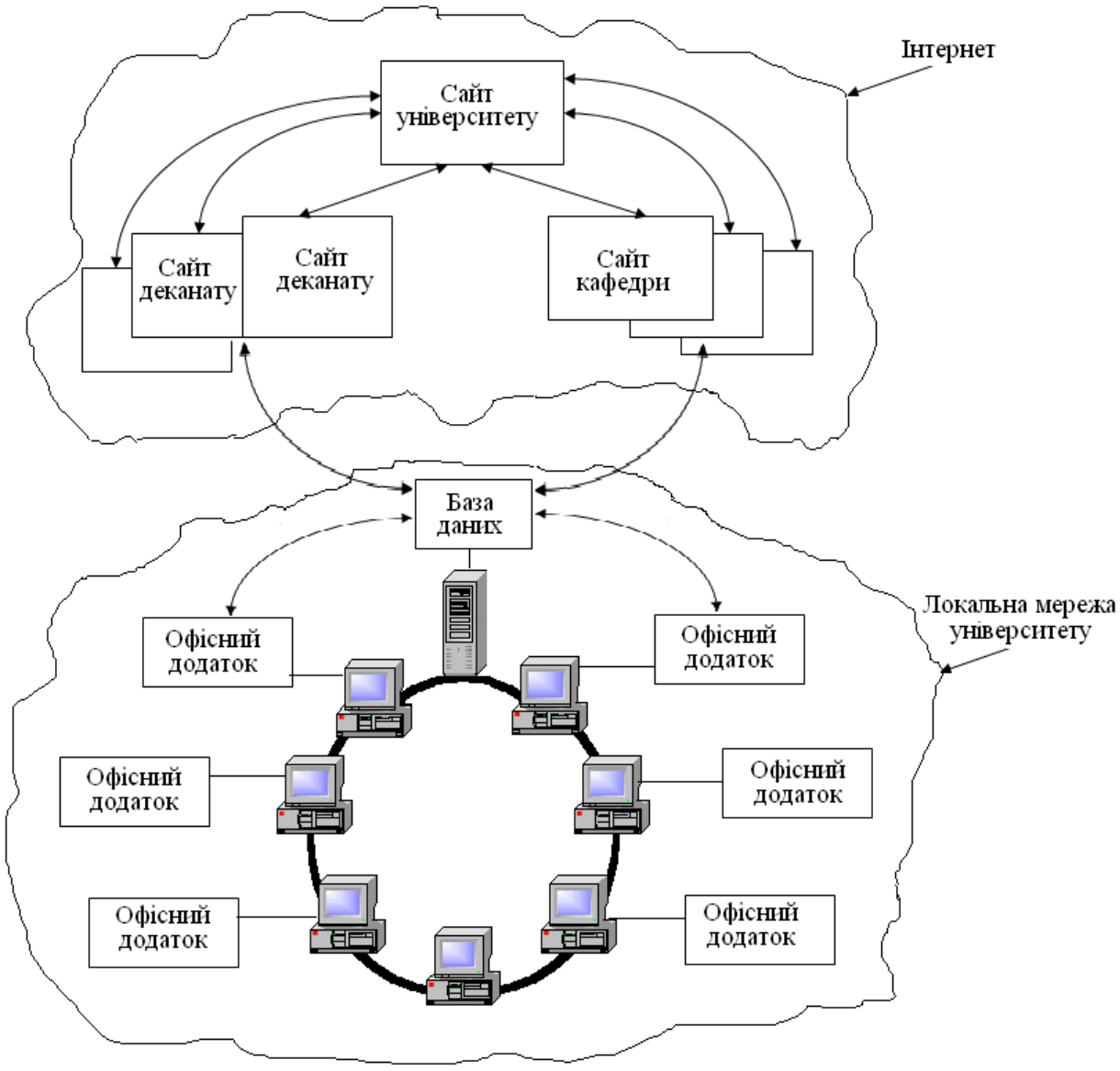

Рисунок 1.1 – Структура ІС «Університет»

Основний сайт університету не повинен мати зв’язок з базою даних для забезпечення максимальної швидкості роботи сайту. Сайти кафедр та деканатів повинні мати зв’язок з базою даних, наприклад, для отримання користувачами сайту розкладу занять академічних груп або викладачів університету. Офісні застосування дозволяють оновлювати інформацію в базі даних, а також переглядати потрібну інформацію в потрібному форматі різним групам авторизованих користувачів.

Серед користувачів ІС можна виділити наступні групи:

1) Гості – можуть тільки переглядати заданий обсяг інформації на сайтах університету.
2) Студенти – можуть переглядати інформацію на сайтах, у тому числі графіки контролюючих заходів по дисциплінах, оцінки по модулях, інтегральні відомості власної успішності, розклад занять. За допомогою офісного застосування «Тестуюча система» студенти можуть виконувати різноманітні тести різної складності та різного призначення: тести для самоперевірки знань та тести, що входять до набору модульних контрольних робіт. Тести для самоперевірки знань студентами можуть бути включені у сайти кафедр або факультетів, але можливості офісних застосувань для реалізації різноманітних комп’ютерних тестів дещо ширші, ніж можливості веб-застосувань.
3) Викладачі – можуть переглядати інформацію на сайтах, у тому числі навчальні плани факультетів, графіки контролюючих заходів та робочі програми по власних дисциплінах, оцінки по модулях, інтегральні відомості академічних груп, розклади занять академічних груп та викладачів. За допомогою сайту кафедри можуть оновлювати інформацію по робочим програмам власних дисциплін, оцінкам по модулям. За допомогою офісного застосування для викладача можуть оновлювати ту саму інформацію, а також заповнювати базу даних тестуючої системи по власних дисциплінах.
4) Співробітники університету – можуть переглядати інформацію на сайтах, а за допомогою відповідних офісних застосувань повинні оновлювати в базі даних ту інформацію, за яку вони відповідають, наприклад, навчальні плани факультету для користувачів «Диспетчер деканату» або «Декан».

Раніше на кафедрі інформаційних технологій у якості дипломних проектів розроблялись ІС «Електронний деканат», «Тестуюча система», «Інтегральні відомості». Ці системи мають однакові групи користувачів, а також ба-

гато спільних типів сутностей в базах даних, а саме: СТУДЕНТИ, ГРУПИ, ВИКЛАДАЧІ, НАВЧАЛЬНІ ПЛАНИ, РОБОЧІ ПРОГРАМИ, ДИСЦИПЛІНИ, тощо. При створенні ІС «Університет» необхідно інтегрувати бази даних вказаних локальних ІС у єдину базу даних.

Під час інтегрування інформаційних систем виникає також проблема інтеграції додатків. Існуючі програмні застосування можуть бути легко перероблені, якщо вони розроблялись у вигляді «тонкого клієнта», що надають користувачеві необхідний інтерфейс для роботи, але не містять у своєму коді процедур обробки інформації – обробка виконується на сервері БД. В цьому випадку нову версію додатка до бази даних потрібно буде тільки підключити до нової версії БД. Надалі всі програмні застосування ІС «Університет» потрібно розробляти саме у вигляді «тонкого клієнта», а для нової обробки даних поповнювати набір збережених процедур – серверне програмне забезпечення ІС. Використання в клієнтських застосуваннях тільки представлень та збережених процедур БД забезпечує виконання вимоги незалежності даних.

2 СИСТЕМНИЙ АНАЛІЗ І КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Етапи проектування бази даних

Процес проектування бази даних складається з трьох основних етапів: концептуальне, логічне і фізичне проектування.

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 [2,10]:

- 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.

2.2 Концептуальне проектування бази даних

2.2.1 Визначення основних типів сутностей

В університеті навчальні дисципліни і викладачі приписані до кафедр. Академічні групи відносяться до факультетів і курсів; кожна група вчиться по своїх навчальних планах, зі своїми дисциплінами, але декілька груп в якомусь семестрі можуть вчиться по одному і тому ж навчальному плану. По дисциплінах можуть передбачатися заняття різного виду: лекції, практичні заняття (семінари), лабораторні заняття.

Викладачі, які ведуть дисципліну, складають робочу програму. У цій робочій програмі вказуються теми, які розглядаються в дисципліні, а також вказуються контролюючі заходи і оцінки за них у вигляді балів. З оцінок за

виконання контролюючих засобів розраховується накопичувальна оцінка поточної успішності студента з дисципліни.

Контролюючі заходи можуть бути у вигляді комп'ютерних тестів і некомп'ютерних завдань: контрольних робіт, рефератів, колоквіумів і т.д. Оцінка за виконання комп'ютерних тестів виставляється автоматично по завершенню студентом тесту, некомп'ютерні контролюючі заходи викладач повинен перевіряти вручну і виставляти оцінку сам.

У робочій програмі вказуються змістовні модулі, на які розподіляється вивчення дисципліни. Один модуль містить одну або декілька тем, та може включати в себе один або декілька контролюючих заходів. Оцінки за окремі контролюючі заходи модулю підсумовуються, і сумарна оцінка за виконання кожного змістовного модулю виставляється у інтегральну відомість поточної успішності студента з дисципліни.

Таким чином, можна виділити основні типи сутностей в схемі бази даних:

- Факультет;
- Кафедра;
- Група;
- Навчальний план;
- Студент;
- Викладач;
- Дисципліна (предмет);
- Робоча програма;
- Модуль;
- Контроль (контролюючий захід);
- Оцінка (студента по контролюючому заходу).

Між вказаними типами сутностей існують наступні типи зв’язків. Факультет має декілька академічних груп та декілька навчальних планів – між типами сутностей Факультет та Група існує тип зв’язку «один до багатьох»; так саме, як і між типами сутностей Факультет та Навчальний_план.

У групі навчається декілька студентів, кожен студент відноситься лише до однієї групи – між типами сутностей Група та Студент існує тип зв’язку «один до багатьох».

Кожен навчальний план містить декілька дисциплін, кожна дисципліна може зустрічатись у декількох навчальних планах. У загальному випадку між типами сутностей Навчальний_план та Дисципліна існує тип зв’язку «багато до багатьох». Хоча на етапі концептуального проектування допускаються рі-

зноманітні типи зв’язків між типами сутностей, але у контексті бази даних, що розглядається, бажано вже на цьому етапі ввести похідний тип сутності Пункт_плану, тому що робочі програми і контролюючі заходи по дисципліні пов’язані не лише з самою дисципліною, але й з навчальним планом, за яким вона вивчається, тобто з пунктом навчального плану.

Навчальний план має декілька пунктів – між типами сутностей Навчальний_план та Пункт_плану існує тип зв’язку «один до багатьох»; так саме, як і між типами сутностей Дисципліна та Пункт_плану.

Для кожного пункту навчального плану провідний викладач складає одну робочу програму – між типами сутностей Робоча_програма та Пункт_плану існує тип зв’язку «один до одного». Кожна робоча програма з дисципліни містить декілька змістовних модулів – між типами сутностей Робоча_програма та Модуль існує тип зв’язку «один до багатьох». Один модуль може включати в себе один або декілька контролюючих заходів – між типами сутностей Модуль та Контроль існує тип зв’язку «один до багатьох». По кожному контролюючому заходу викладач ставить декілька оцінок (різним студентам), та у кожного студента є декілька оцінок (по різним контролюючим заходам) – між типами сутностей Контроль та Оцінка існує тип зв’язку «один до багатьох»; так саме, як і між типами сутностей Студент та Оцінка.

На кожній кафедрі працює декілька викладачів, а також до кафедри приписуються декілька навчальних дисциплін – між типами сутностей Кафедра та Викладач існує тип зв’язку «один до багатьох»; так саме, як і між типами сутностей Кафедра та Дисципліна.

Виявивши типи зв'язків між виявленими типами сутностей, отримуємо основну ER-діаграму підсхеми «Успішність студентів» БД «Університет» (рисунок.2.1).

Контролюючий захід може бути комп'ютерним тестом. На кафедрі інформаційних технологій вже є база даних «Тестуюча система», яку потрібно інтегрувати в інформаційну систему «Університет».

У базі даних «Тестуюча система» питання тестів відносяться до якої-небудь теми, а кожна тема відноситься до якої-небудь дисципліни. Для кожного питання повинно бути кілька варіантів відповіді, з яких один або кілька є правильними.

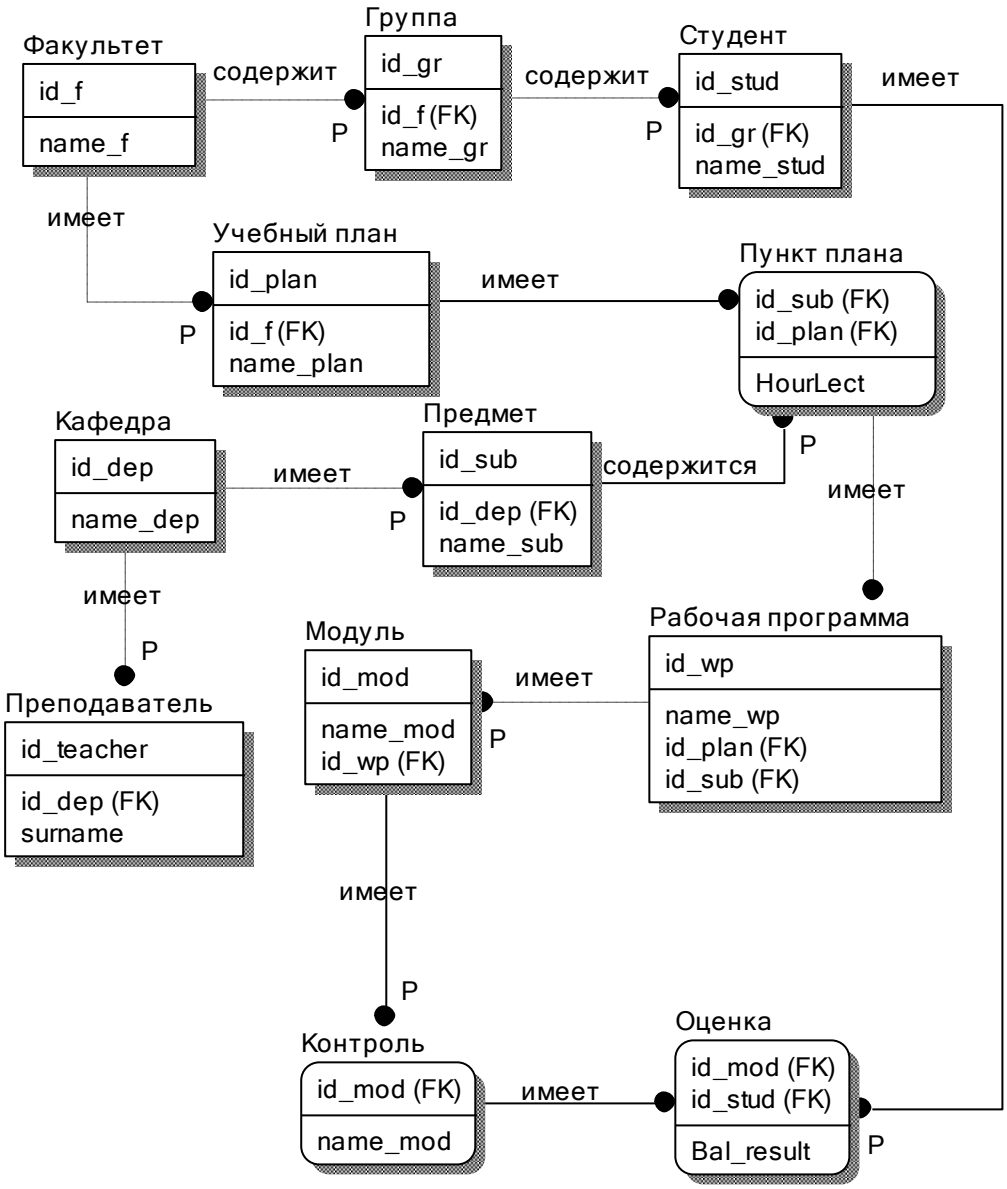

Рисунок 2.1 – Основна ER-діаграма підсхеми «Успішність студентів» бази даних ІС «Університет»

Також є питання, для яких зберігаються не різні відповіді на питання, а фрагменти відповіді. У якості правильної відповіді на таке питання зберігається послідовність фрагментів, що становлять правильну відповідь на запитання.

Для кожного комп'ютерного тесту потрібно задати список тем, які входять в тест, і кількість питань по кожній темі. Конкретні питання з кожної теми для кожного студента при тестуванні програма вибирає випадковим чином. Студент дає відповідь на кожне питання тест. З балів, отриманих за відповіді на питання, виходить оцінка за тест.

Основними типами сутностей бази даних «Тестуюча система» є:

- Предмет (навчальна дисципліна);
- Тема;
- Питання;
- Відповідь;
- Правильна відповідь;
- Студент;
- Тест;
- Склад теста (теми, що включені в тест);
- Відповідь студента на питання;
- Оцінка.

Визначимо типи зв’язків, що існують між виявленими типами сутностей.

Між типами сутностей Предмет і Тема існує тип зв’язку «один до багатьох».

Між типами сутностей Тема і Питання існує тип зв’язку «один до багатьох».

Між типами сутностей Питання і Відповідь існує тип зв’язку «один до багатьох».

Між типами сутностей Відповідь і Правильна_відповідь існує тип зв’язку «один до багатьох».

Між типами сутностей Предмет і Тест існує тип зв’язку «один до багатьох».

Між типами сутностей Питання і Відповідь_на_питання існує тип зв’язку «один до багатьох».

Між типами сутностей Тест і Відповідь_на_питання існує тип зв’язку «один до багатьох».

Між типами сутностей Студент і Відповідь_на_питання також існує тип зв’язку «один до багатьох».

Оцінка студента за тест розраховується з оцінок за кожне питання, що входять до складу тесту. Між типами сутностей Студент і Оцінка також існує тип зв’язку «один до багатьох» – оцінки студента за різні тести.

Між типами сутностей Тест і Оцінка також існує тип зв’язку «один до багатьох» – оцінки різних студентів за тест.

Описав типи зв’язків між виявленими типами сутностей, отримуємо ER-діаграму підсхеми «Тести» БД «Університет» (рис. .2.2).

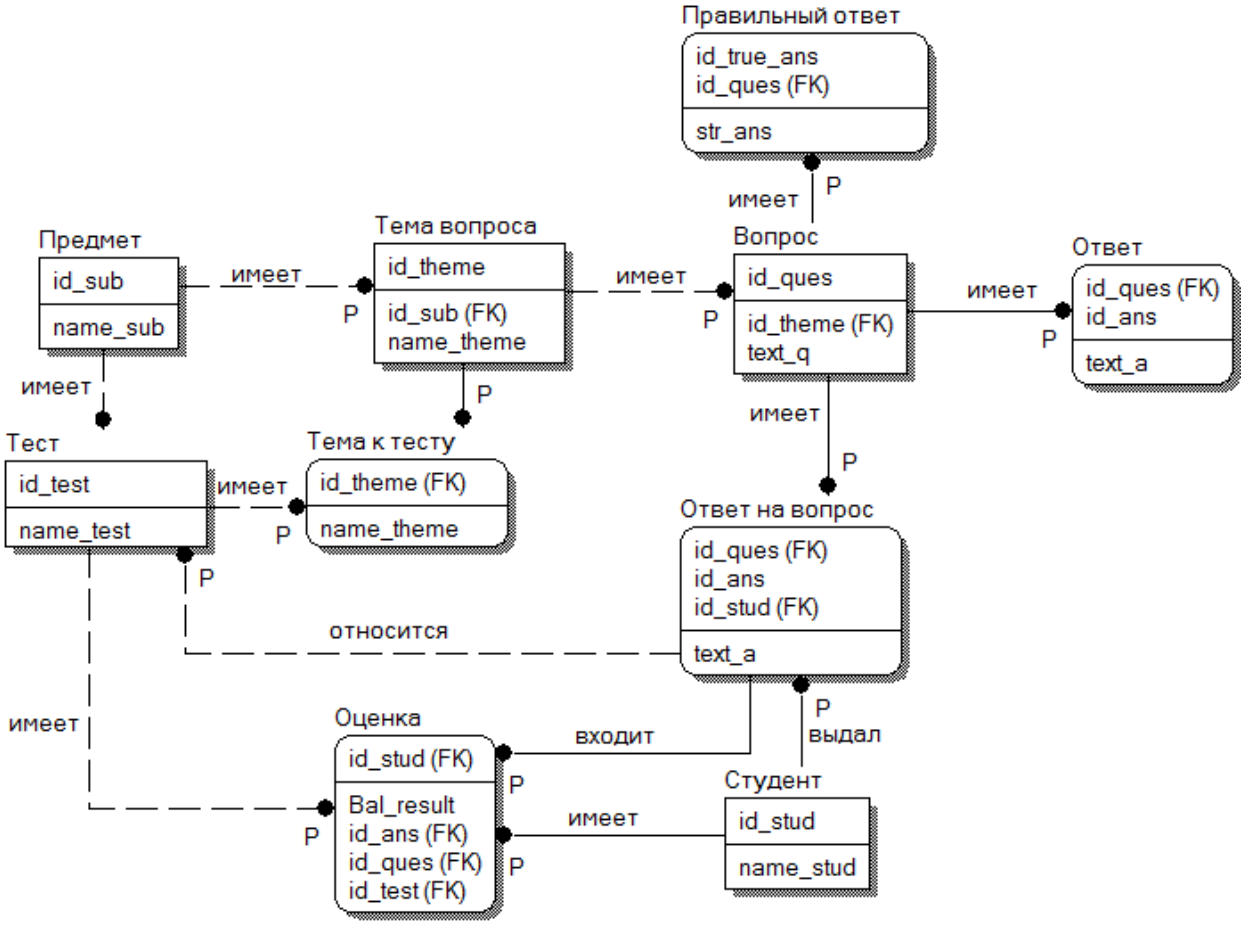

Рисунок 2.2 – ER-діаграма підсхеми «Тести»

Наведені підсхеми потрібно об'єднати в одну базу даних з урахуванням того, що тести є одним з видів контролюючих заходів з дисциплін.

Таким чином отримуємо ER-діаграму підсхеми «Успішність студентів» бази даних ІС «Університет» (рис. 2.3).

Дана підсхема інформаційної системи дозволяє вести контроль поточної успішності студентів, в тому числі з її допомогою можна генерувати інтегральні відомості різного виду: по одній дисципліні всім студентам однієї групи або з усіх дисциплін для одного студента.

Для інтеграції БД «Електронний деканат» у інформаційну систему «Університет» слід пам’ятати, що в розглянутій вище підсхемі «Успішність студентів» вважається, що в базі даних зберігається інформація про поточну успішність, тобто ця схема описує процес навчання студентів протягом одного семестру. В базі даних «Електронний деканат» повинні зберігатись відомості про весь процес навчання студента в університеті. Також в ній потрібні зберігатись навчальні плани за минулі роки, та перспективні навчальні плани на майбутнє.

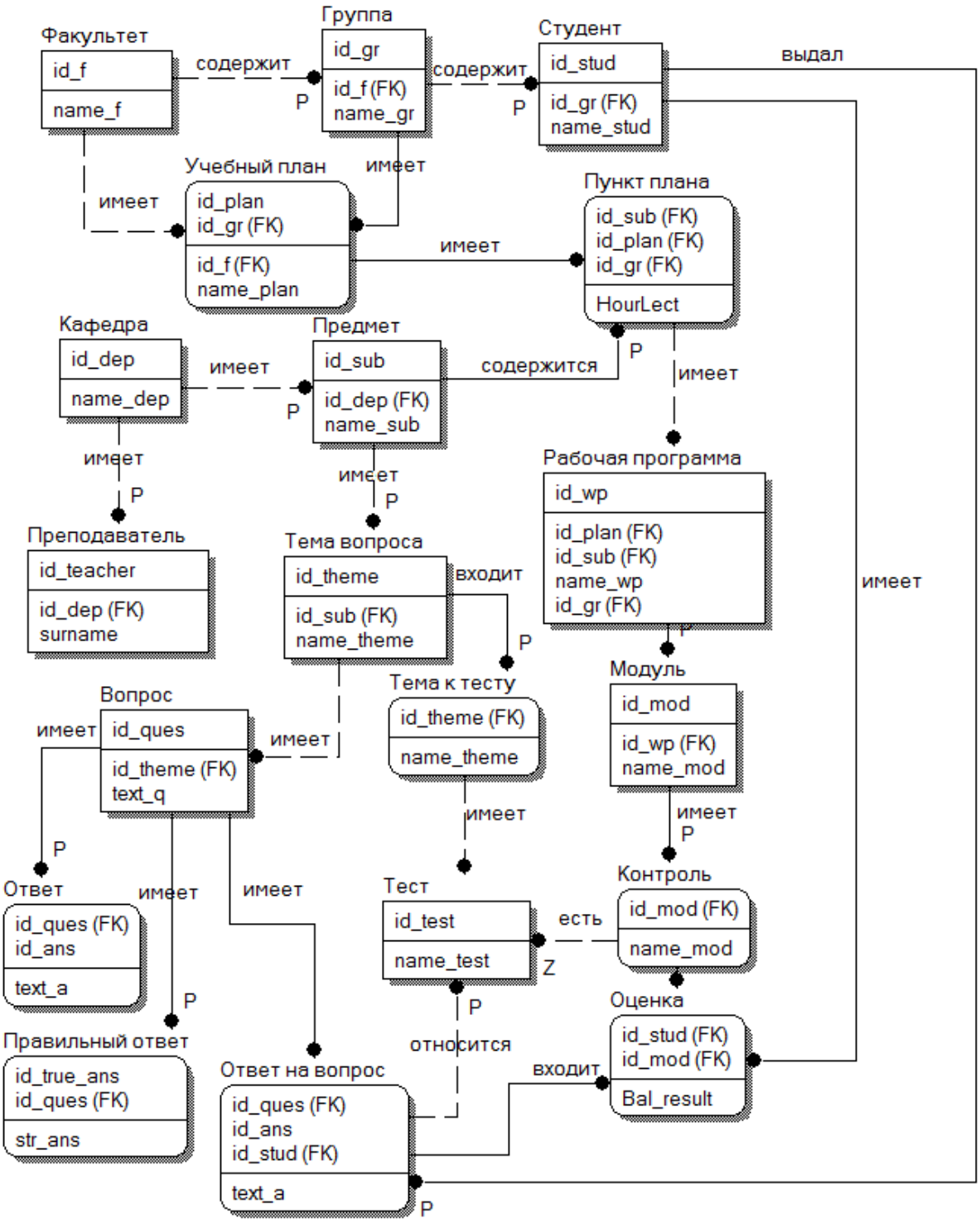

Рисунок 2.3 – Повна ER-діаграма підсхеми «Успішність студентів» бази даних ІС «Університет»

Для зберігання множини навчальних планів, які відносяться до різних навчальних років, не потрібно вносити зміни у ER-діаграму на рис. 2.3, потрібно лише додати атрибут «навчальний рік» у перелік атрибутів типу сутності Навчальний_план.

Але для можливості зберігання історії вивчення студентом навчальних дисциплін на протязі всіх років потрібно додати новий асоціативний тип сутності Плани_Студента, в якому в кожному навчальному році кожному студенту ставиться у відповідність навчальний план, за яким він навчався.

Можна також замість типу сутності Плани_студента додати новий асоціативний тип сутності Групи_студента, в якому кожному студенту в кожному навчальному році ставиться у відповідність академічна група, у якій він навчався. У цьому випадку додатково потрібно буде додати новий асоціативний тип сутності Плани_групи, в якому кожній групі в кожному навчальному році ставиться у відповідність навчальний план, за яким вона навчалась. Цей підхід хоча у призведе до деякої надмірності даних у базі даних, але значно спростить можливість розрахунків різних статистичних показників успішності академічних груп. Тому потрібно вибрати саме його.

Але введення асоціативного типу сутності Плани_групи не відміняє необхідність існування асоціативного типу сутності Плани_студента. Цей тип сутності краще ввести у схему замість асоціативного типу сутностей Групи_студента, тому що така схема дозволить простіше реалізувати можливість зберігання у базі даних відомостей про індивідуальні навчальні плани студентів.

За індивідуальним навчальним планом студент має право виключати для себе деякі навчальні дисципліни з навчального плану групи, в якій він навчається, так звані «Дисципліни вільного вибору студента».

Замість видалених вибіркових дисциплін студент має право додавати у свій індивідуальний навчальний план будь-які інші навчальні дисципліни з інших навчальних планів, навіть з навчальних планів для інших спеціальностей. Звичайно, заміна однієї навчальної дисципліни на іншу може вважатись не рівноцінною, тому для введених студентом у індивідуальний план додаткових дисциплін призначаються вагові коефіцієнти, значення яких, як правило, є дробовою величиною менше 1. На цей ваговий коефіцієнт множиться кількість кредитів навчальної дисципліни, яку додав до свого індивідуального навчального плану студент.

Введення у схему бази даних двох таблиць: таблиці виключення студентом дисципліни з навчального плану та таблиці додавання студентом дисципліни з іншого навчального плану, – повністю описує індивідуальні навчальні плани студентів.

Остаточна ER-діаграма підсхеми «Електронний деканат та успішність студентів» бази даних ІС «Університет» приведена на рис. 2.4.

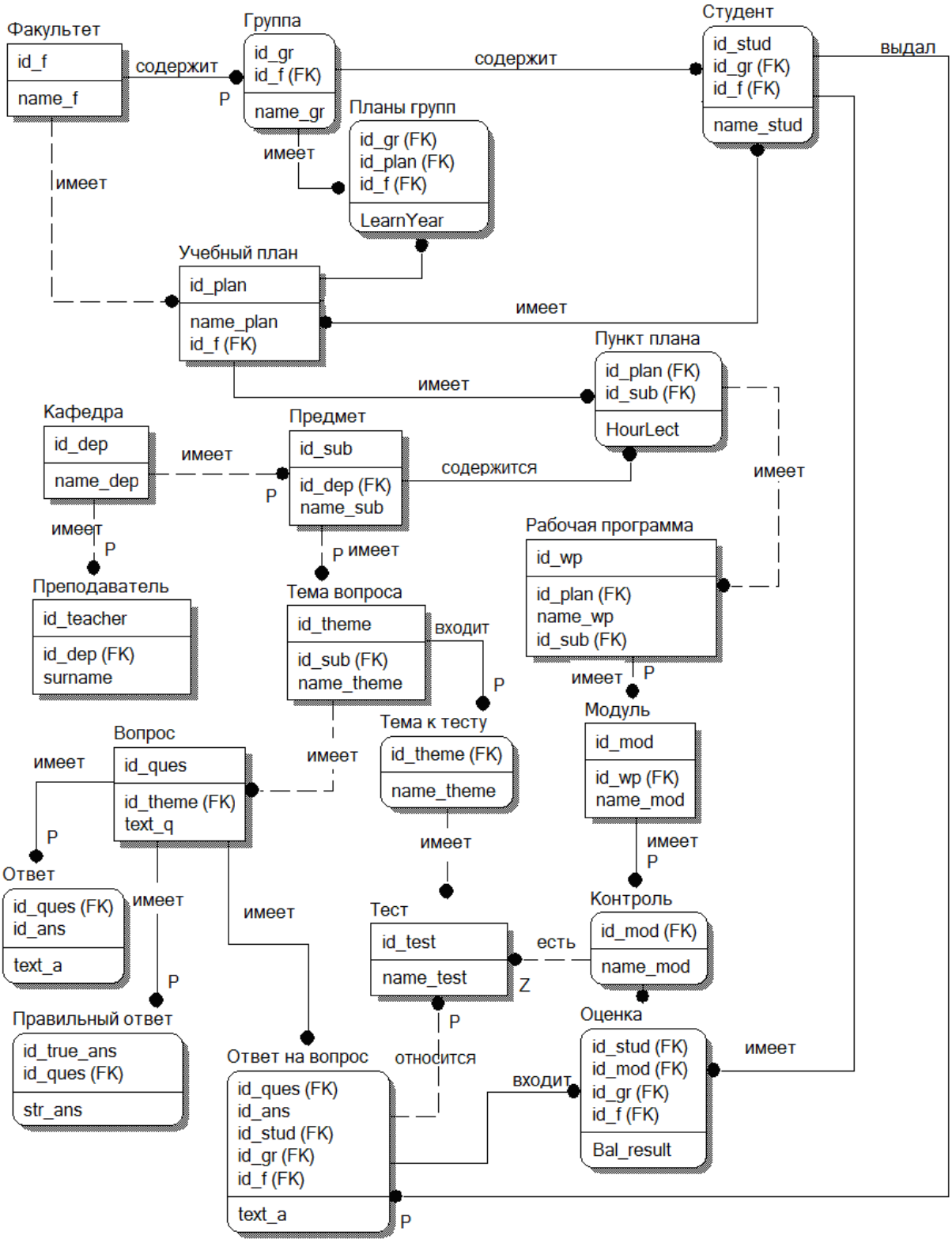

Рисунок 2.4 – Остаточна ER-діаграма підсхеми «Електронний деканат та успішність студентів» бази даних ІС «Університет»

Як видно з рис. 2.4, між типами сутностей є один тип зв’язків «багато до багатьох», що недопустимий для реляційної моделі даних. Тому логічна

модель БД, що розробляється, для реляційної моделі даних буде дещо відмінною від розробленої концептуальної моделі бази даних.

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

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

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

У типу сутності Група будуть атрибути: назва (name_gr), код групи (id_gr), код факультету (id_f).

У типу сутності Навчальний_план будуть атрибути: назва (name_plan), код навчального плану (id_plan), код факультету (id_f), навчальний рік (LearnYear).

У типу сутності Студент будуть атрибути: код студента (id_stud), номер залікової книжки (number), прізвище (surname), ім’я (name1), по-батькові (name2).

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

У типу сутності Предмет будуть атрибути: код предмета (id_sub), код кафедри (id_dep), назва (name_sub), коротка назва (short_name_sub).

У типу сутності Пункт_плану будуть атрибути: код навчального плану (id_plan), код предмета (id_sub), кількість кредитів (credit), кількість годин лекцій (HourLect), кількість годин семінарів (HourPract), кількість годин лабораторних занять (HourLab), кількість годин самостійної роботи (HourOwn).

У типу сутності Викладач будуть атрибути: код викладача (id_teacher), код кафедри (id_dep), код посади (id_pos), прізвище (surname), ім’я (name1), по батькові (name2), величина ставки (id_user).

У типу сутності Робоча_програма будуть атрибути: код програми (id_wp), назва (name_wp), код пункту навчального плану (id_plan).

У типу сутності Модуль будуть атрибути: код модулю (id_mod), назва (name_mod), код програми (id_sub).

У типу сутності Контроль будуть атрибути: код модулю (id_modWP), назва (name_modWP), вид контролю (чи є контролюючий захід комп’ютерним тестом) (KindModul), максимальний бал (MaxBal), дата здачі за графіком (DateReport).

У типу сутності Оцінка будуть атрибути: код студента (id_stud) , код контролю (id_modWP), дата здачі (Date_mod), оцінка у відсотках (Bal_result) .

У типу сутності Тема (предмета) будуть атрибути: код теми (id_theme), назва (name_theme), код предмета (id_sub).

У типу сутності Тема_теста будуть атрибути: код теми теста (id_theme), назва (name_theme), код теми предмета (id_sub), категорія (різновид) питань (id_cat), максимальний бал за питання з теми (Amount_mark).

У типу сутності Питання будуть атрибути: код питання (id_ques), текст питання (text_q), код теми теста (id_theme), кількість правильних відповідей (kol_true), рисунок до питання (pict).

У типу сутності Відповідь будуть атрибути: код питання (id_ques), номер відповіді (id_ans), текст відповіді (text_a), рисунок до відповіді (pict_ans).

У типу сутності Правильна Відповідь будуть атрибути: код питання (id_ques), текст правильної відповіді (text_a_rtf).

У типу сутності Тест будуть атрибути: код теста (id_test), назва (name_test), код предмета (id_sub), тривалість у хвилинах (Length_min).

У типу сутності Склад_теста будуть атрибути: код теста (id_test), код теми теста (id_theme), кількість питань з теми у одному варіанті теста (kol).

У типу сутності Відповідь_студента будуть атрибути: код студента (id_stud), код питання (id_ques), номер відповіді (id_ans), дата та час відповіді (DateTest).

Більшість виявлених атрибутів відносяться або до домену імен (назв): прізвище, ім’я, по батькові, назва предмета, назва факультету, назва дисципліни, – або до доменна цілих чисел: номер (код) предмета, код кафедри, кількість студентів в групі. Атрибут «чи є комп’ютерним тестом» відноситься до логічного типу. Атрибути «рисунок до питання» та «рисунок до відповіді» відноситься до типу даних image. Вказані типи даних допустимі для СУБД, заснованих на будь-якої з сучасних моделей даних.

2.2.3 Визначення відповідності концептуальної моделі транзакціям користувачів

Для перевірки відповідності концептуальної моделі вимогам призначених для користувача транзакцій, визначимо, які транзакції можуть знадобитися всім групам користувачів, а саме: диспетчерам деканатів та деканам, викладачам, студентам.

Диспетчеру деканату потрібні наступні транзакції:

- переглянути список академічних груп факультету;
- переглянути контингент груп факультету;
- перевести студентів не останнього курсу на наступний курс в началі навчального року;
- перевести окремого студента у іншу групу;
- зачислити нового студента у будь-яку групу.

Декану потрібні наступні транзакції:

- переглянути список академічних груп факультету;
- переглянути контингент груп факультету;
- переглянути навчальні плани;
- відригувати навчальний план або додати новий;
- поставити академічній групі у відповідність навчальний план, за яким вона буде навчатись наступного року;
- переглянути графіки контролюючих заходів по дисциплінам, що вивчаються за планами факультету;
- отримати інтегральні відомості поточної успішності студентів факультету.

Викладачеві потрібні наступні транзакції:

- переглянути список академічних груп факультету;
- переглянути контингент груп факультету;
- переглянути навчальні плани;
- оновити робочі програми по дисциплінах;
- скласти графік контролюючих заходів у відповідності до робочої програми по дисципліні;
- заповнити базу даних комп’ютерних тестів по дисципліні (теми, питання, відповіді, тести);
- отримати оцінки за виконані студентами комп’ютерні тести по дисципліні;
- поставити оцінки за виконання студентами некомп’ютерних контролюючих заходів;
- отримати інтегральні відомості поточної успішності студентів по дисципліні.

Студенту потрібні наступні транзакції:

- переглянути навчальний план, за яким навчається група поточного року;

- переглянути графіки контролюючих заходів по всіх дисциплінах, що повинен вивчати студент у поточному семестрі;
- переглянути оцінки, отримані студентом за будь-який контролюючий захід;
- отримати інтегральні відомості поточної успішності по всіх дисциплінах, що вивчає студент у поточному семестрі.

Для усіх цих транзакцій визначені сутності, їх атрибути і зв’язки між ними. Отже, отримана концептуальна модель коректна.

3 ВИБІР СУБД І СЕРЕДОВИЩА РОЗРОБКИ КЛІЄНТСЬКОГО ДОДАТКУ

3.1 Вибір СУБД

Для нашого проекту буде раціонально залишатися працювати з SQL-сервером замість MySQL з тієї причини, що у нас буде веб-додаток та офісний додаток. Тому вибирати за основу MySQL який більше направлений для розробки веб-сайтів немає сенсу, тим більше, що для веб-сайтів використовують такі додатки для розробки баз даних як C ++ Builder, Visual Studio на різні мови. Крім того, в MySQL є велика бібліотека уявлень і збережених процедур, які абсолютно нераціонально буде заново переписувати під MySQL, враховуючи, що MySQL для створення офісних додатків практично не використовується. Інформаційна система, що розробляється, буде використовуватися і для сайтів, наприклад, сайти кафедр і факультетів, але все ж краще буде залишити СУБД Microsoft SQL Server. Розробляти додатки до неї можна в будь-якому середовищі (C ++ Будера, Delphi, Visual Studio) і на будь-якій мові (C ++, C #, Visual Basic).

Система SQL Server 2008 відштовхується від концепції платформи даних Майкрософт: вона спрощує управління будь-якими даними в будь-якому місці і в будь-який момент часу. Вона дозволяє зберігати в базах даних різноманітну інформацію, отриману з структурованих, напівструктурованих і неструктурованих джерел, таких як зображення і музика. У SQL Server 2008 є великий набір інтегрованих служб, що розширюють можливості використання даних: ви можете складати запити, виконувати пошук, проводити синхронізацію, робити звіти, аналізувати дані. Всі дані зберігаються на основних серверах, що входять до складу центру обробки даних. До них здійснюється доступ з настільних комп'ютерів і мобільних пристроїв. Таким чином, ви повністю контролюєте дані незалежно від того, де ви їх зберегли.

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

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

Microsoft SQL Server 2008 – це надійна, продуктивна і інтелектуальна платформа даних, здатна відповідати потребам найбільш ресурсномістких бізнес-додатків. Вона дозволяє скоротити час і витрати на розробку і супровід додатків, а також надавати практично застосовну інформацію на кожне робоче місце підприємства.

SQL Server 2008 R2 містить низку нових функцій, що дозволяють організації впевнено масштабувати операції з базами даних, підвищити продуктивність праці IT-фахівців і розробників, а також впровадити добре масштабується і кероване рішення для бізнес-аналізу без використання програмування.

SQL Server 2008 має вбудовану функцію стиснення, що дозволяє стискати файли бази даних і журналів транзакцій, пов'язані з відповідною базою даних. У SQL Server 2005 з'явилася можливість стиснення даних файлів або файлових груп, доступних тільки для читання, але ця форма стиснення просто використовує можливість стиснення файлової системи Windows NTFS. У SQL Server 2008 з'явилося стиснення на рівні рядків і сторінок, яке надає переваги, недоступні при стисненні на рівні файлів даних.

Стиснення на рівні рядків і сторінок зменшує необхідний простір даних, а також необхідний об'єкт пам'яті, оскільки дані в пам'яті залишаються стислими. Стислі дані в пам'яті дозволяють збільшити використання пам'яті, що вигідно для масштабованості безлічі систем.

У SQL Server 2008 також з'явилося стиснення на рівні резервної копії. Хоча при резервному копіюванні бази даних створюється тільки резервна копія активної частини бази даних, це забезпечує вивільнення сотень гігабайт або навіть десятків терабайт вільного простору. У середовищах баз даних з декількома копіями файлу резервної копії обсягом кілька терабайт ці резервні копії часто займають цінне дисковий простір, яке могло використовуватися більш ефективно. Надаючи адміністраторам баз даних можливість стиснення файлів резервних копій SQL Server 2008 звільняє частину цього простору, щоб воно могло використовуватися для реальних даних.

Також з'явився регулятор ресурсів. Ця нова функція дозволяє визначати обсяг ресурсів, який може використовуватися окремими робочими навантаженнями або групами робочих навантажень під час виконання. Регулятор ресурсів дозволяє створювати середовище, в якому на одному сервері співіснує безліч різних робочих навантажень без небезпеки того, що одна або кілька цих навантажень викличуть перевантаження сервера або зниження продуктивності інших робочих навантажень.

Перевага цієї функції полягає в тому, що вона забезпечує можливість більш ефективного використання загального обсягу ресурсів, доступних на серверах баз даних.

SQL Server 2008 збільшує загальну продуктивність баз даних. Завдяки кільком новим функціям, представленим в SQL Server 2008, з'явилася можливість керування і спостереження за продуктивністю баз даних і додатків, які виконуються на них.

При наявності безлічі транзакцій, що виконуються кожну секунду, блокування, зазвичай відбувається під час виконання цих транзакцій, може мати негативний вплив на продуктивність додатків бази даних. SQL Server забезпечує зменшення загального число блокувань, які зберігаються процесом, завдяки укрупненню невеликих блокувань рівня рядків і сторінок до великих блокувань рівня таблиць. Але важливо розуміти, що таке укрупнення блокувань може викликати проблеми. Наприклад, одна транзакція може блокувати всю таблицю і перешкоджати роботі інших транзакцій з нею.

SQL Server 2008 працює з механізмом секціонування таблиць (який з'явився в SQL Server 2005), дозволяючи ядру SQL Server укрупнювати блокування на рівні секції до рівня таблиці. Цей проміжний рівень блокування може значно знизити ефект укрупнення блокувань в системах з обробкою сотень і тисяч транзакцій в секунду.

SQL Server 2008 пропонує кілька нових поліпшень обробника запитів для взаємодії запиту з секціонованими таблицями. Тепер оптимізатор запитів може виконувати пошук запитів в секціях так само, як і в окремих індексах, працюючи тільки з ідентифікатором секції, а не з механізмом секціонування на рівні таблиць.

З підвищенням складності середовищ баз даних і зростанням розмірів баз даних можливість забезпечення доступності цих баз даних значно ускладнюється. Знайомі механізми, що використовувалися раніше для досягнення високої доступності, збереглися в SQL Server 2008. Але деякі з цих функцій поліпшені в SQL Server 2008, крім того, додані нові функції.

Сервер SQL Server 2005 дозволив безлічі адміністраторів використовувати дзеркальне відображення бази даних для досягнення високої доступності. SQL Server 2008 надає безліч поліпшень дзеркального відображення бази даних. Наприклад, раніше дзеркальне відображення бази даних іноді призводило до появи проблем продуктивності, що відносяться до перенесення даних журналу транзакцій з основних баз даних у дзеркальні. З урахуванням цього SQL Server 2008 тепер зменшує обсяг інформації, переданої по мережі між журналами транзакцій основний і дзеркальної баз даних, за допомогою стиснення інформації до її відправки в журнал транзакції дзеркальної бази даних з метою посилення захисту.

Тепер є можливість відновлення пошкоджених сторінок даних на основному сервері. Якщо в основній базі даних є пошкоджені сторінки даних, обумовлені помилками 823 і 824, основний сервер може запросити поточну копію цих сторінок даних у дзеркальних серверів. Цей запит робочих сторінок даних – автоматизований процес, який прихований це всіх користувачів, що звертаються в цей час до основних баз даних.

Інша нова функція, ЦП з підтримкою гарячої заміни, дозволяє встановлювати на сервері бази даних додаткові ЦП, не впливаючи на доступність баз даних, що знаходяться на цьому сервері. Однак необхідно знати, що ЦП з підтримкою гарячої заміни має деякі обмеження, оскільки корисна тільки для 64-розрядних випусків Windows Server 2008 Enterprise Edition і Datacenter Edition для систем з процесорами Itanium і для неї необхідний випуск Enterprise Edition сервера SQL Server 2008.

Служба SQL Server Master Data Services забезпечує узгодженість даних в неоднорідних системах, технологія SQL Server StreamInsight ефективно обробляє складні потоки событій.Платформа підтримує можливість вертикального масштабування на потужних сучасних серверах архітектури x64 і Itanium до 256 логічних процесорів.

Адміністратори можуть централізовано відстежувати і контролювати кілька серверів, примірників і додатків баз даних, прискорюючи розробку і розгортання додатків і забезпечуючи поліпшену підтримку віртуалізації за допомогою технології Hyper-V з функцією динамічної міграції зі складу Windows Server 2008 R2.

Надбудова SQL Server PowerPivot для Excel дає всім користувачам доступ до розвинених засобам бізнес-аналізу. Бізнес-користувачі можуть створювати і поширювати рішення для бізнес-аналізу самостійно або з мінімальною підтримкою IT-фахівців, в той час як IT-відділ зберігає за собою функції

спостереження та управління рішеннями для бізнес-аналізу, які були створені користувачами.

Microsoft SQL Server 2008 дозволяє створювати комплексні корпоративні рішення в області аналітики, що дають цінну практичну інформацію про діяльність компанії. При роботі використовуються тільки добре знайомі користувачеві інструменти.

Microsoft SQL Server 2008 допоможе в масштабуванні ресурсномістких аналітичних додатків з мільйонами рядків даних і тисячами користувачів.

Єдина модель UDM задовольняє будь аналітичним потребам, від багатовимірного аналізу та створення звітів до прогнозуючої аналітики, – все в рамках єдиного рішення.

Засоби упреждающей аналітики Microsoft SQL Server 2008, засновані на потужному інтелектуальному аналізі даних і тісно інтегровані з платформою Microsoft BI, дозволяють приймати зважені, обґрунтовані рішення. Їх можливості будуть доступні для будь-якої програми.

Microsoft SQL Server 2008 підвищує продуктивність праці розробників і дозволяє легко створювати надійні рішення баз даних нового покоління. Це досягається за рахунок потужного мови програмування T-SQL і багатого набору типів даних, здатних зберігати практично що завгодно.

Нові і поліпшені можливості мови T-SQL в SQL Server 2008 дозволяють створювати додатки баз даних нового покоління, які відповідають найбільш суворим вимогам користувачів. Покращена підтримка типів даних в SQL Server 2008 дозволяє працювати з реляційними і нереляціоннимі даними, у тому числі з точними свідченнями дати і часу, даними XML, зовнішніми документами і файлами, а також нової просторовою інформацією в площинному або геодезичному поданні. Зміни в ядрі зберігання SQL Server і його сховище даних дозволяють ефективно зберігати, маніпулювати і шукати дані за допомогою розріджених стовпців, відфільтрованих індексів і нового, повністю інтегрованого повнотекстового пошуку.

У SQL Server 2008 є функції, що підвищують ефективність управління параметрами систем безпеки, суворої перевірки автентичності та контролю доступу, шифрування та управління ключами, розширеного аудиту.

СУБД, спроектована як безпечна за замовчуванням і безпечна в розгортанні, дозволить підвищити захищеність даних. Доступ до даних можна надавати тільки тим, кому це дійсно потрібно. Використовуються засоби ефективного управління перевіркою достовірності та авторизацією. Захист конфіденційної інформації забезпечується вбудованими криптографічними засо-

бами і підтримкою корпоративного управління ключами. Дії, які здійснюються в базі даних, можна піддавати аудиту. Це дозволяє створювати необхідну звітність і відповідати нормативним вимогам.

3.2 Вибір середовища розробки кліентського додадтку до БД

Однією з перших вдалих розробок в області розробки додатків до баз даних є програмне середовище Borland C ++ Builder.

Компанією Microsoft були розроблені власні пакети для швидкої розробки додатків, у тому числі додатків для баз даних. Сама вдала їх розробка в цій області – це пакет .NET разом із середовищем розробки додатків Visual Studio.

Існуючі клієнтські програми інформаційної системи «Тестуюча система» розроблені в середовищі Borland C ++ Builder. Однак при створенні набору додатків для однієї інформаційної системи немає необхідності використовувати одну й ту ж програмне середовище або один і той же мова програмування. Кліенскіе додатки можуть створюватися за допомогою будь-якого середовища швидкої розробки додатків для баз даних.

Середовище розробки Microsoft Visual Studio існує вже більше 8 років на ринку і в даний час входить в число найпопулярніших інструментів для розробки корпоративних додатків і веб-інтерфейсів з використанням .NET. У 2010 році світ побачила Microsoft Visual Studio 2010, що змінила попередню версію Microsoft Visual Studio 2008. Що нового з'явилося в цьому продукті? Про це в нашому матеріалі

Системні вимоги, підготовка до роботи

IDE Microsoft Visual Studio 2010 випущена для роботи на комп'ютерах під управлінням ОС Windows 2003/XP/Vista/ 008/Windows 7 з відповідними встановленими оновленнями. Для установки буде потрібно близько 7.5 Гб вільного дискового простору. Також у системі повинен бути встановлений браузер IE8, бібліотека .NET Framework і пакет MS Office 2007 або 2010.

Основні компоненти

Будь-який проект з розробки сьогодні вимагає інтерактивної взаємодії всіх його учасників – як виконавців, так і керівників, і замовників. Для цього в середовищі розробки повинні підтримуватися інструменти для спільної роботи – централізоване сховище інформації з гнучким механізмом розмежування доступу до контенту, наочні інструменти стану проекту і внеску працівника в досягнення ключових показників, якісну систему відстеження змін,

вже внесених в проект і очікуваних для застосування. В MS Visual Studio 2010 за це відповідає компонент Team Foundation Server. З його допомогою групи осіб мають доступ до єдиного сховища вимог, які своєчасно оновлюються і актуалізуються. В інтерфейсі є інструменти, що дозволяють показувати актуальний зріз вимог проекту, а також будувати звіти і таблиці по ходу проекту. Крім того, вся ця інформація оновлюється в режимі реального часу, що дозволяє уникнути ситуації, коли внесені зміни і пропозиції можуть «загубитися», терміни виконання – «порушитися», а вже виконані вимоги продубльовані розробниками.

Інший компонент середовища розробки – Visual Studio Lab Management 2010 дозволяє швидко будувати діаграми частин проекту – спочатку залежностей для поточного рішення, щоб отримати уявлення про те, як воно працює, а потім вже і послідовності дій для нових функцій, які з'являться в новому проекті. Всі елементи можуть бути легко створені безпосередньо з частин діаграм із збереженням загального дизайну проекту в незмінному вигляді. У Microsoft Visual Studio 2010 застосовується успішна технологія UML-моделювання, що включає відповідні описи класів і компонентів, що входять до бази самої IDE. Всього в продукті використовується кілька типів UML-діаграм:

– діаграми діяльностей;

– діаграми варіантів використання;

– діаграми послідовностей;

– діаграми класів;

– діаграми компонентів.

Таким чином наочно можна побачити не тільки всі зв'язки об'єктів коду, а також помилки зв'язків, а й «вузькі» місця, які необхідно оптимізувати.

Наступний етап створення проекту зазвичай пов'язаний з появою вже прототипу нового рішення, оформленого, наприклад, у вигляді графічного дизайну програми. Для об'єднання творчих потенціалів дизайнерів і розробників в Microsoft Visual Studio 2010 використовується спеціальний компонент MS Expression 3 / Expression Blend, до складу якого входить інструмент SketchFlow. Зовні він виглядає як редактор, в якому можна створювати розширені медіарісунки, що містять не просто графічне оформлення частин і деталей проекту, але й концептуальну їх складову (зв'язку, навігацію, управління, форми і так далі). Цифрові прототипи в результаті виглядають як інтерактивні зображення, які забезпечені реальними елементами програмного коду майбутньої реалізації проекту, до яких розробники можуть залишати свої по-

значки, зауваження та пропозиції через Team Foundation Server, де ці файли і розміщуються (при цьому сам процес публікації спрощений до буквально одного кліка). При бажанні доступ до проекту можна організувати і через веб-інтерфейс, який не потребує інсталяції пакета. Також слід згадати про можливість перегляду ресурсів безпосередньо з MS SharePoint 2010.

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

Робота з існуючими проектами

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

Природно, що існуючі пертурбації вимагають тестування на предмет забезпечення повного переходу між версіями. Що з'явилося в продукті вікно Test Impact View відображає всі впливу змін в коді на тестування проекту –

розробник зможе побачити, які тести йому потрібно виконати після того чи іншого впровадження або виключення фрагмента, перемикаючись швидко між самим кодом і списком тестів. У підсумку стає досить просто відстежити, які саме розробники допускають помилки, в чому вони полягають і наскільки вони критичні. В Team Foundation Server 2010 з'явилося відповідне засіб, який дозволяє керівнику проекту реєструвати надходять від програміста зміни в коді – відповідно, якщо вони прийняті або не прийняті, він отримує про це повідомлення. Таким чином зменшується до мінімуму ризик збоїв при складанні, якщо проблема виникла на якомусь з етапів. Крім усього іншого, є і спеціальні інструменти налагодження і профілювання створених багатопоточних додатків.

У Microsoft Visual Studio останньої версії помітно спростилися і покращилися в плані користувальницького інтерфейсу інструменти візуалізації коду – дизайнери і програмісти бачать звичний кожним з них інтерфейс (виконаний з використанням Windows Presentation Foundation і Silverlight; для підвищення зручності сприйняття, зокрема, з нього прибрані деякі лінії і градієнти прибрані, щоб знизити захараченість панелей), при цьому завжди є можливість перемикання між режимами. З істотних нововведень варто відзначити підтримку мультимоніторних систем – це зручно для налагодження коду. Крім того, з вікна середовища розробки можна завантажувати, встановлювати, спільно використовувати і управляти шаблонами, пакетами і компонентами.

В IDE залишилася і отримала новий розвиток система IntelliTrace, яка підвищує продуктивність налагодження коду за рахунок фіксації стану програми не тільки в певний момент часу, але реєструє їх на певному періоді до і після налагодження, відповідно, дозволяючи виявити і помилки, і причини, чому вона відбувається. При перегляді подій, що мали місце в ході виконання програми, розробникам доступний весь контекст налагодження, включаючи вікно перегляду значень, стек викликів, вікно інтерпретації і вікно точок зупинки. Крім того, Microsoft оптимізувала саму по собі IntelliTrace, знизивши до 2-5 разів швидкість синтаксичного розбору коду будь-якого розміру.

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

лізацію розробки. Крім того, Microsoft Visual Studio 2010 сумісна з фірмовою «хмарної» платформою Azure.

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

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

Поліпшення існуючих технологій. У платформі .net є технології, які не змінилися радикально в даному випуску, але були прибрані багато помилок і була проведена деяка доробка. До таких технологій відноситься технологія ASP.NET.

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

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

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

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

Модульне тестування та розробка, керована тестами – середовище розробки visual studio 2010 включає в себе велику кількість розширень, що допомагають у реалізації цих стратегій. Так само з'явилися нові динамічні засоби і середовище DLR, а так само архітектура ASP.NET MVC.

Використання ручного тестування.

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

Управління проектом з урахуванням майбутнього.

Visual Studio 2010 оптимізована для сучасного ітеративного процесу розробки і оснащена функціями збереження продуктивності та усунення потенційних проблем до їх появи. Продукт дозволяє стежити за працездатністю проекту, використовуючи автоматично створювані звіти. Крім того, можна управляти завантаженням проекту за допомогою даних журналів і документів планування на основі Microsoft Excel.

Управління життєвим циклом додатків (ALM).

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

Налагодження і діагностика.

У Visual Studio 2010 Ultimate представлена IntelliTrace, корисна функція налагодження, яка дозволяє залишити всі проблеми з відтворюваністю помилок у минулому. Тестери можуть детально і ефективно описувати помилки і розробники завжди зможуть відтворити їх у тому стані, в якому їх виявили. До інших можливостей відносяться статичний аналіз коду, метрика коду та профілювання.

Засоби тестування.

Visual Studio 2010 Ultimate оснащена всіма удосконаленими засобами тестування для забезпечення стабільної якості коду. Скористайтеся кодованими тестами інтерфейсу користувача, за допомогою яких можна автоматизувати тестування інтерфейсу веб-додатків і додатків Windows, засобами ручного тестування, функцією Test Professional, тестуванням продуктивності веб-додатків, тестуванням навантаження, охоплення коду та іншими корисними функціями, які недоступні в інших версіях Visual Studio.

Розробка баз даних.

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

З наведеного огляду можливостей середи швидкої розробки додатків Visual Studio 2010 можна зробити висновок, що ця середа – наймасштабніший за всю історію цього середовища розробки реліз Microsoft за кількістю та значимістю реалізованих нововведень. Рішення відрізняється поліпшеним інтерфейсом і наявністю нових інструментів, що дозволяють автоматизувати виконання рутинних операцій, а також підтримки актуальних технологій (Windows 7, багатоядерних систем через багато-додатками в .NET Framework 4, «хмари» Windows Azure і т.д.) і значно вдосконаленого механізму командної розробки.

4 ЛОГІЧНЕ І ФІЗИЧНЕ ПРОЕКТУВАННЯ БД

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

4.1 Логічне проектування

Розроблена в другому розділі модель вже майже є логічною моделлю реляційної бази даних. Як видно з ER-моделі, представленої на рис. 2.4, ніде немає розривів зв’язків, але є зв’язок типу «багато до багатьох» між типами сутностей Навчальний_план і Студент

Для відповідності логічної моделі БД вибраній реляційній моделі даних необхідно ввести асоціативні сутності, які дозволять перетворити один тип зв’язку «багато до багатьох» в два типи зв’язку «один до багатьох». Тому вводиться асоціативний тип сутності План_студенту. У цього типу сутності будуть атрибути: код студенту, код навчального плану, навчальний рік – які визначають навчальний план, за яким вчився студент у кожному навчальному році.

Таким чином, набір сутностей, визначених при концептуальному проектуванні, перетвориться 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 практично співпадає з концептуальною моделлю, для якої вказана перевірка була виконана.

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

Отримаємо ER-діаграмау логічної моделі «Електронний деканат та успішність студентів» БД «Університет», яка показана на рис. 4.1 та у додатку А.

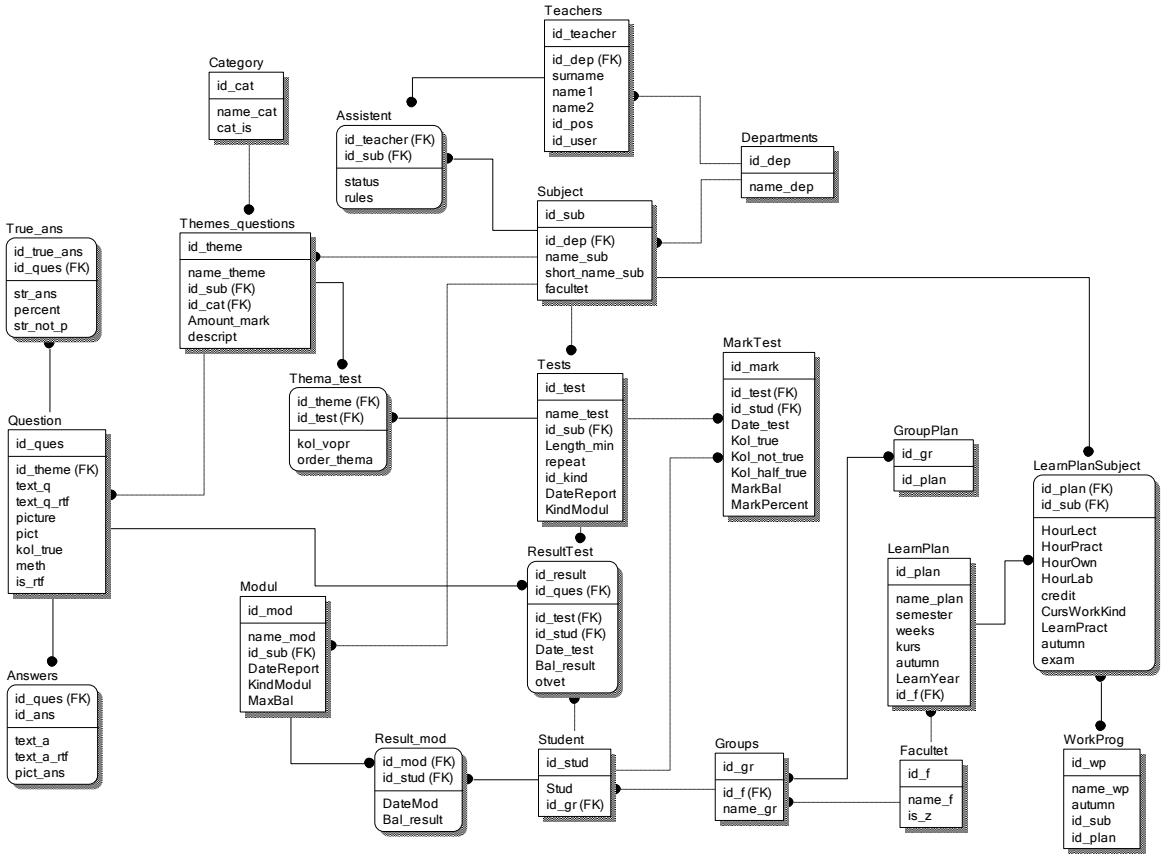

Рисунок 4.1 – ER-диаграмма логічної моделі бази даних «Університет»

Логічна модель БД наведена у додатку Б.

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

Таблиця 4.1 – Імена відношень бази даних «Університет»

Відношення	Назва англійською
Факультет	Facultet
Група	Groups
Предмет	Subject
Навчальни план	LearnPlan
Дисципліна (Предмет)	LearnPlan Subject
Категорія	Category
Кафедра	Departments
Асистент	Assistant

Продовження таблиці 4.1

Відношення	Назва англійською
Викладач	Teachers
Тест	Test
Тема тесту	Thema_test
Тема питання	Themes_questions
Питання	Question
Відповідь на питання	Answers
Правильна відповідь	True_ans
Модуль	Modul
Оцінка за модуль	Result_mod
Студент	Student
Рабоча програма	WorkProg
План групи	GroupPlan
Оцінка за тест	MarkTest
Результат теста	ResultTest

4.2 Фізичне проектування

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

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

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

Для таблиць, вміст дані поступають з декількох джерел: списки викладачів кафедр, списки навчальних дисциплін, списки академічних груп, навчальні плани, – бажано використати обчислювані первинні ключі, значення яких відповідає джерелу, з якого отримані ці дані. Наприклад, первинний ключ в таблиці Група – код групи – обчислюється залежно від коду (номеру) факультету, до якого відноситься група; так само залежно від

коду факультету повинний обчислюватися код навчального плану, тобто первинний ключ в таблиці Навчальний_план.

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

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

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

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

N – кількість біт в записі значення code.

Для обчислення ключів вимагається визначити розмірність значення code для кожної таблиці і вичислити розмірність усіх ключів.

У таблицях Група і Навчальний_план первинні ключі мають тип smallint, тобто є знаковими цілими числами розміром в два байти.

Зазвичай на факультеті не буває більше 20 навчальних планів, але можна із запасом відвести 6 біт на запис номера навчального плана усередині факультету – тобто навчальних планів на факультеті не повинно бути більше 63.

Також у таблиці Groups первинний ключ має тип smallint. Бітова маска для визначення первинного ключа таблиці LearnPlan (Навчальний план) показана на рис. 4.2.

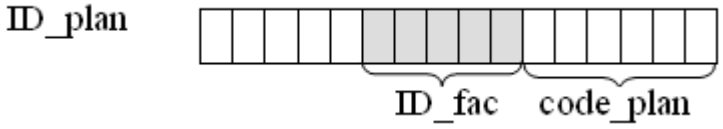

Рисунок 4.2 – Розподіл бітів запису зовнішнього ключа і порядкового номера в запису первинного ключа таблиці LearnPlan

Таблиця LearnPlanSubject підпорядкована двом таблицям: LearnPlan і Subject. Первинний ключ таблиці LearnPlanSubject повинен складатись з комбінації її зовнішніх ключів, тобто ID_sub та ID_plan.

Відповідно до логічної схеми, спроектованої в розділі 4.1 і приведеної в додатку А, для вибраної СУБД MS SQL Server 2000 можна спроектувати фізичну базу даних. Для цього для кожного атрибуту треба вказати домен, тобто область допустимих значень, і відповідно до домена вибрати найбільш відповідний тип для атрибуту.

Також до фізичної моделі бази даних відносяться представлення бази даних. SQL-код представлень БД наведений у додатку В.

5 СЕРВЕРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

У базі даних «Тестуюча система» має бути серверне ПЗ у вигляді набору збережених процедур, що забезпечують маніпулювання даними в таблицях бази даних: додавання записів, оновлення, видалення, а також для необхідну обробку даних.

Процедура LearnPlanAdd додає запис в таблицю LearnPlan:

```
create procedure LearnPlanAdd
@id_f tinyint,
@name_plan varchar (50),
@id_plan smallint output,
@semester tinyint,
@weeks tinyint,
@z_f  bit,
@k tinyint output,
@msg varchar (50) output
```

Параметри процедури:

@id_f – код факультету;

@name_plan – назва навчального плану;

@semester – номер навчального семестру;

@weeks – кількість навчальних тижнів в семестрі за планом;

@id_plan – код плану, який вираховується як первинний ключ таблиці, вихідний параметр;

@z_f – чи є факультет заочним;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура LearnPlanAdd обчислює останній зайнятий номер навчального плану всередині факультету (від 1 до 63). Якщо цей номер менше 63, то додається запис про навчальний план з первинним ключем, який на 1 більше максимального коду навчального плану на цьому факультеті. Якщо в таблиці є запис, для якої id_plan% 64 = 63, то процедура знаходить мінімальний вільний номер навчального плану на факультеті (від 1 до 62). Якщо всі номери зайняті, тобто на факультеті є 63 навчальних плану, то генерується повідомлення про помилку.

Процедура LearnPlanDel видаляє запис з таблиці LearnPlan:

```
create procedure LearnPlanDel
@id_plan smallint,
```

@m tinyint,

@k tinyint output,

@msg varchar (200) output

Параметри процедури:

@id_plan – код навчального плану;

@m – параметр, що визначає дії процедури при наявності записів у підпорядкованій таблиці;

@k – вихідний параметр, кількість записів в підлеглих таблицях;

@ msg – вихідний параметр, повідомлення про помилку.

Якщо параметр @m > 0, то відразу з підлеглих таблиць видаляються записи зі значенням атрибута id_plan, рівного @id_plan. Потім видаляється запис з таблиці LearnPlan зі значенням атрибута id_plan, рівного @id_plan

Якщо параметр @ m = 0, то для навчального плану зі значенням id_plan, рівним @id_plan, знаходяться кількість записів в підлеглих таблицях. Ця інформація повертається через параметр @msg і навчальний план не видаляється. Якщо у підпорядкованих таблицях немає записів із зовнішнім ключем id_plan = @id_plan, з таблиці LearnPlan видаляється запис з таким значенням первинного ключа id_plan.

Процедура LearnPlanEdit оновлює запис у таблиці LearnPlan:

create procedure LearnPlanEdit

@id_plan smallint,

@name_plan varchar (50),

@semester tinyint,

@weeks tinyint,

@k tinyint output,

@msg varchar (50) output

Параметри процедури ті ж самі, що і в процедурі LearnPlanAdd.

Для навчального плану, код якого дорівнюєі @d_plan, процедура LearnPlanEdit привласнює нові значення атрибутів name_plan, weeks, semester. Якщо навчальний план з такою назвою вже є на факультеті, генерується повідомлення про помилку.

Процедура LearnPlanSubAdd додає в таблицю LearnPlanSubject запис:

create procedure LearnPlanSubAdd

@id_plan smallint,

@name_sub varchar(80),

@HourLect smallint,

@HourPract smallint,

@HourLab smallint,

@k tinyint output,

@msg varchar(100) output

Параметри процедури:

@id_plan – код навчального плану;

@name_sub – коротка назва навчальної дисципліни;

@HourLect – кількість годин лекцій на тиждень з дисципліни;

@HourPract – кількість годин практичних занять на тиждень з дисципліни;

@HourLab – кількість годин лабораторних робіт на тиждень з дисципліни;

@k – код помилки, параметр що повертається;

@msg – повідомлення про помилку, параметр що повертається.

Процедура LearnPlanSubAdd перевіряє, чи є дисципліна з короткою назвою, рівним значенню @name_sub, у списку дисциплін університету на поточний семестр. Якщо ні – повертається повідомлення про помилку. Якщо така дисципліна є, то в таблицю LearnPlanSubject додається запис про наявність у плані з кодом @id_plan дисципліни з короткою назвою @name_sub з кількістю годин лекцій, практичних (семінарських) занять і лабораторних занять, відповідних параметрам @HourLect, @HourPract, @HourLab.

Процедура LearnPlanSubEdit призначена для оновлення записів в таблиці LearnPlanSubject:

create procedure LearnPlanSubEdit

@id_plan smallint,

@id_sub_old smallint,

@id_sub_new smallint,

@HourLect smallint,

@HourPract smallint,

@HourLab smallint,

@k tinyint output,

@msg varchar(100) output

Параметри процедури:

@id_plan – код навчального плану;

@id_sub_old – попередній код навчальної дисципліни;

@id_sub_new – новий код навчальної дисципліни;

@HourLect – кількість годин лекцій на тиждень з дисципліни;

@HourPract – кількість годин практичних занять на тиждень з дисципліни;

@HourLab – кількість годин лабораторних робіт на тиждень з дисципліни;

@k – код помилки, параметр що повертається;

@msg – повідомлення про помилку, параметр що повертається.

Процедура LearnPlanSubEdit дозволяє оновити кількість годин занять на тиждень: лекцій, практичних занять, лабораторних занять, – з дисципліни з кодом @id_sub_old в навчальному плані з кодом @id_plan. Також процедура дозволяє замінити в навчальному плані з кодом @id_plan дисципліну з кодом @id_sub_old на дисципліну з кодом @id_sub_new. При зміні назви дисципліни кількість годин занять також можна змінити. Якщо нова дисципліна вже є в даному навчальному плані, то повертається повідомлення про помилку.

Процедура LearnPlanSubDel видаляє записи з таблиці LearnPlanSubject:

create procedure LearnPlanSubDel

@id_plan smallint,

@short_name_sub varchar(15),

@m tinyint,

@k tinyint output,

@msg varchar(100) output

Параметри процедури:

@id_plan – код навчального плану;

@short_name_sub – коротка назва навчальної дисципліни;

@m – параметр, що визначає дії процедури при наявності записів у підпорядкованій таблиці;

@k – вихідний параметр, кількість записів в підпорядкованих таблицях;

@ msg – вихідний параметр, повідомлення про помилку.

Якщо параметр @m > 0, то відразу з підлеглих таблиць видаляються записи зі значенням атрибута id_plan, рівного @id_plan для дисципліни з короткою назвою @short_name_sub. Потім видаляється запис з таблиці LearnPlanSubject зі значенням атрибута id_plan, рівного @id_plan і short_name_sub = @short_name_sub.

Якщо параметр @m = 0, то обчислюється кількість записів у підпорядкованій таблиці GroupStudy зі значенням атрибута id_plan, рівного @id_plan для дисципліни з короткою назвою @short_name_sub. Якщо таких

записів немає, то з таблиці LearnPlanSubject видаляється запис зі значенням атрибута id_plan, рівного @id_plan і short_name_sub = @short_name_sub.

Якщо записи в підпорядкованій таблиці є, то повертається повідомлення про помилку.

Процедура GroupPlanAddEdit додає і оновлює записи в таблиці GroupPlan:

create procedure GroupPlanAddEdit

@id_gr smallint,

@id_plan smallint

Параметри процедури:

@id_gr – код групи;

@id_plan – код навчального плану.

Процедура GroupPlanAddEdit визначає, чи приписаний до групи з кодом @id_gr якийсь навчальний план. Якщо план не приписаний, то в таблицю GroupPlan додається запис з атрибутами id_gr = @id_gr, id_plan = @id_plan. Якщо план приписаний, то він замінюється на план з кодом @id_plan. Якщо за попереднім навчальним планом в базі даних були заняття, то вони видаляються.

Процедура GroupPlanDel видаляє запис з таблиці GroupPlan:

create procedure GroupPlanDel

@id_gr smallint,

@m tinyint,

@k tinyint output,

@msg varchar (200) output

Параметри процедури описані вище.

Дії процедури аналогічні діям процедур GroupPlanDel і LearnPlanDel.

Процедура GroupPlanFill додає і оновлює записи в таблиці GroupPlan:

create procedure GroupPlanFill

@id_plan smallint,

@id_gr smallint

Параметри процедури:

@id_gr – код групи;

@id_plan – код навчального плану, за яким вчиться група в поточному семестрі.

Процедура GroupPlanFill зчитує з таблиці LearnPlanSubject пункти навчального плану: коротка назва навчальної дисципліни, кількість годин лекцій, кількість годин практичних занять, кількість годин лабораторних

занять, відповідні навчальним планом з кодом @id_plan. З цих пунктів навчального плану формуються заняття для групи з кодом @id_gr. Одне заняття групи в базі даних означає заняття тривалістю 2 години по одному виду (лекція, практичне або лабораторне) по одній дисципліни. Якщо по якомусь виду у якоїсь дисципліни буде в навчальному плані непарна кількість годин (1, 3, 5 і т.д.), то одне із занять цього виду з цієї дисципліни в базі даних отримає позначку, що воно проводиться через тиждень. Наприклад, для групи з деякою дисципліни в навчальному плані передбачено 3:00 лекцій. У базі даних сформується 2 заняття у вигляді лекцій з цієї дисципліни: одне заняття буде з відміткою, що воно проводиться щотижня, інше заняття буде з відміткою, що воно проводиться через тиждень.

Сформовані заняття записуються в таблицю GroupPlan, яка містить для кожної групи всі заняття, які повинні бути поставлені в розклад. Для можливості переробки і коригування торішнього розкладу в розклад поточного навчального року наявні заняття в таблиці GroupPlan не видаляються, а оновлюються. Наприклад, якщо минулого року у групи за деякою дисципліни була передбачена та ж кількість годин лекцій, що і в цьому році, лекційні заняття в таблиці GroupPlan з цієї дисципліни у цієї групи не зміняться. Якщо торік лекцій з даної дисципліни було менше, то додадуться необхідні заняття, які потім потрібно буде поставити в розклад. Якщо лекцій стало менше, то зайві заняття забираються і з розкладу, і зі списку занять групи (таблиці GroupPlan).

6 ОПИС РОЗРОБЛЕНОГО ДОДАТКУ ДИСПЕТЧЕРУ

6.1 Загальні відомості

Програма розроблена в середовищі розробки додатків Visual Studio 2010, може працювати під управлінням операційної системи Windows NT/ 2000/XP/Vista/7/8 і будь-який інший сумісної мережевою операційною системою. Програма працює як додаток до бази даних, тому в локальній мережі в якій працює програма має бути сервер баз даних із завантаженою базою cafedra01.

6.2 Функціональні призначення

Програма LearnPlan.exe розроблена як офісне застосування у складі інформаційної системи «Університет». Це застосування призначене для груп користувачів «Декан» та «Диспетчер деканату».

Програма дозволяє користувачеві переглядати і оновлювати дані які відносяться до навчальних планів факультетів, контингенту студентів, списків студентів академічних груп. Користувач може працювати лише з даними, які відносяться до його факультету.

6.3 Опис програми

6.3.1 Керівництво програміста

Програма написана у вигляді додатку WinwosFormAplication, складається з 4-х форм: Form1, Form2, Form3, Form4 кожна форма є об'єктом відповідного класу.

Клас основної форми:

public partial class Form1 : Form

Конструктор класу Form1:

public Form1()

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

Методи класу форми Form1:

Методи ініціалізації збережених процедур:

init_commands_LearnPlanDel

Служить для ініціалізації команди виклику збереженої процедури DelLearnPlanCmd

init_commands_LearnPlanSubDel()

Служить для ініціалізації команди виклику збереженої процедури DelLearnPlanSubCmd.

Решта методи класу Form1 є обработчиками подій натискання на кнопки або вибору пункту випадаючого списку.

Обробник події натискання на кнопку AddLpButt:

private void AddLpButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку AddLpButt (додати). Цей метод формує клас параметрів форми Form2, потім створює об'єкт класу Form2 з вказаними параметрами і викликає створену форму.

Обробник події натискання на кнопку EditLpButt:

private void EditLpButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку EditLpButt (змінити). Цей метод формує клас параметрів форми Form2, потім створює об'єкт класу Form2 з вказаними параметрами і викликається створена форма.

Обробник події натискання на кнопку DelLpButt:

private void DelLpButt_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку DelLpButt (видалити). Цей метод викликає збережену процедуру DelLearnPlan для видалення поточного запису в таблиці LearnPlan.

Обробник події вибору пунктів випадаючого списку GroupComboBox:

private void GroupComboBox_SelectedIndexChanged (object sender, EventArgs e)

Цей метод змінній IdGr привласнює значення ідентифікатора групи, відповідного пункту випадаючого списку груп.

Обробник події натискання на кнопку OKButton:

private void OKButton_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку OKButton.

LearnPlanParams inout_params – це клас параметрів форми, через які передаються дані з головної форми на форму Form2.

System.Data.SqlClient.SqlConnection connection – параметр, який з головної форми передає з'єднання з базою даних.

Методи класу форми Form2:

private void init_commands_LearnPlan_add()

Служить для ініціалізації команди виклику збереженої процедури AddPlanCmd.

private void init_commands_LearnPlan_edit()

Служить для ініціалізації команди виклику збереженої процедури EditPlanCmd.

private void OKButton_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку OKButton.

Форма Form3 призначена для додавання та оновлення записів в таблиці LearnPlanSubject. Форма є відповідним об'єктом класу форми.

Клас форми Form3:

public partial class Form3: Form

Конструктор класу Form3:

public Form3 (LearnPlanSubParams inout_params,
System.Data.SqlClient.SqlConnection connection)

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

Параметри конструктора:

LearnPlanSubParams inout_params – це клас параметрів форми, через які передаються дані з головної форми на форму Form3.

System.Data.SqlClient.SqlConnection connection – параметр, який з головної форми передає з'єднання з базою даних.

Методи класу форми Form3:

private void init_commands_LearnPlanSub_add()

Служить для ініціалізації команди виклику збереженої процедури AddPlanSubCmd.

private void init_commands_LearnPlanSub_edit()

Служить для ініціалізації команди виклику збереженої процедури EditPlanSubCmd.

private void OKButton_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку OKButton.

Форма Form4 призначена для додавання та оновлення записів в таблиці Students. Форма є відповідним об'єктом класу форми.

Клас форми Form4:

public partial class Form4: Form

Конструктор класу Form4:

public Form4 (StudentsParams inout_params,
System.Data.SqlClient.SqlConnection connection)

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

Параметри конструктора:

StudentsParams inout_params – це клас параметрів форми, через які передаються дані з головної форми на форму Form4.

System.Data.SqlClient.SqlConnection connection – параметр, який з головної форми передає з'єднання з базою даних.

Методи класу форми Form4:

init_commands_Students_add()

Служить для ініціалізації команди виклику збереженої процедури AddStudentsCmd.

init_commands_Students_edit()

Служить для ініціалізації команди виклику збереженої процедури EditStudentsCmd.

private void OKButton_Click (object sender, EventArgs e)

Метод є обробником події натискання на кнопку OKButton.

6.3.2 Керівництво користувача

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

На першій вкладці можна переглянути наявність наявних навчальних планів по факультетах, як показано на рис. 6.1. Поки користувач нічого не вибрав, він нічого не побачить.

Як видно з рис. 6.1, програма дозволяє переглянути наявність всіх навчальних планів для даного навчального року.

Після вибору диспетчера деканату потрібного йому навчального року в таблиці ліворуч зверху на формі в таблиці ліворуч видно навчальний план (назва плану, навчальний рік, кількість тижнів осіннього семестру та кількість тижнів весіннього семестру),а в таблиці праворуч для вибраного навчального плану видно його склад: дисципліна, кількість годин лекцій, кількість годин практики, кількість годин лабораторних робіт, самостійна робота, курсова робота чи курсовий проект, екзамен, навчальна практика). Понизу вкладки виводиться перелік груп, які навчаються по заданому навчальному плану (назва групи, навчальний рік) (рис. 6.2).

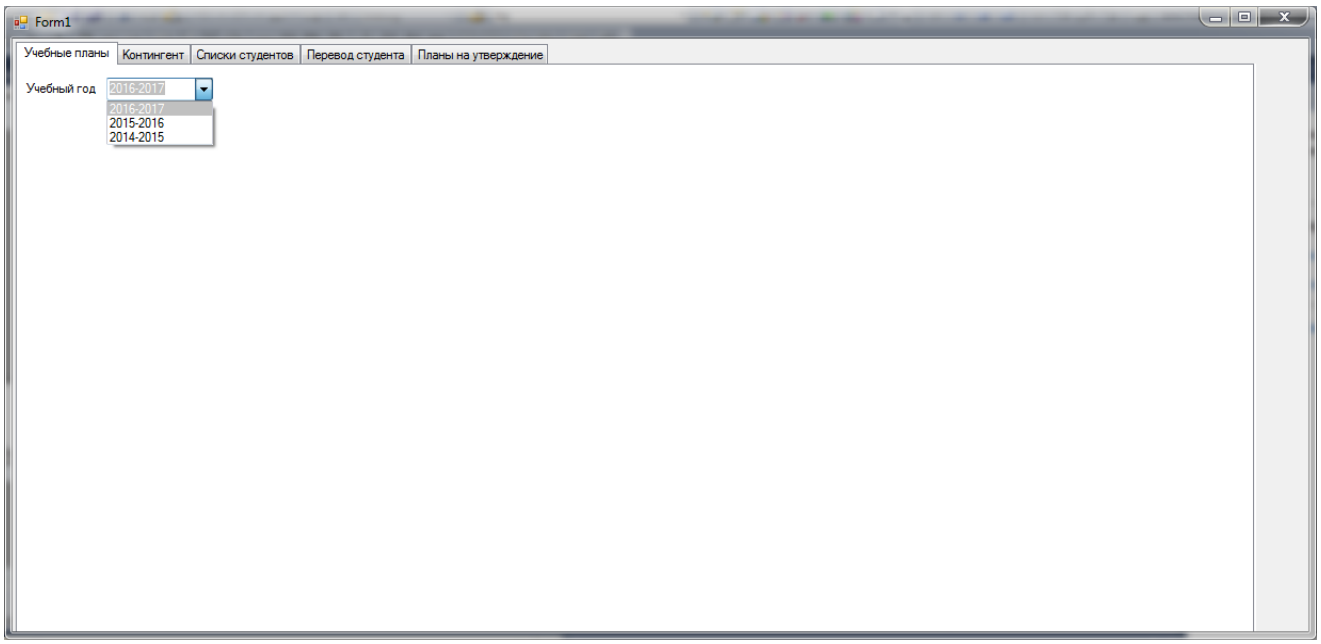

Рисунок 6.1 – Порожня форма навчального плану

Якщо для поточного навчального плану в БД немає його наповнення, тобто переліку навчальних дисциплін, кредитів, кількості годин лекцій, кількості годин практики, кількості годин лабораторних занять то порожнім буде правий табличний компонент.

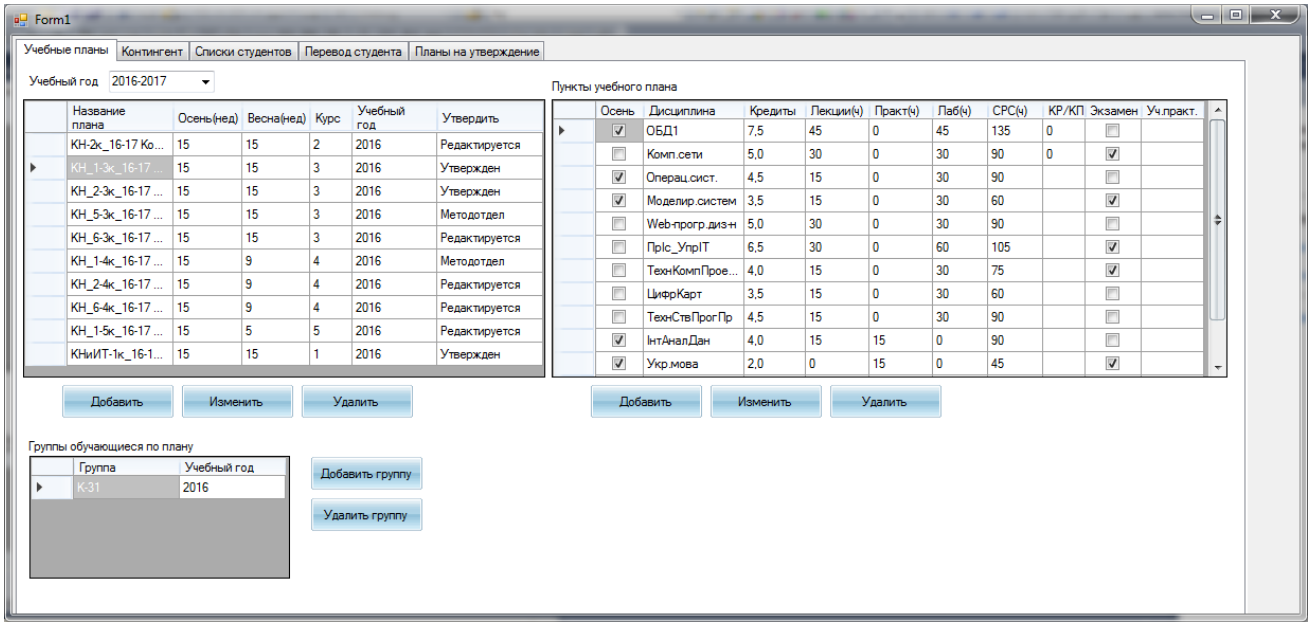

Рисунок 6.2 – Заповнена форма навчального плану

Після натискання диспетчером деканату кнопки додати групу, на головній формі стає доступною прихована панель для додавання нової групи для

вибраного навчального плану. Після того як на формі з'являється прихована панель кнопка додати групу та видалити групу стає недоступною.

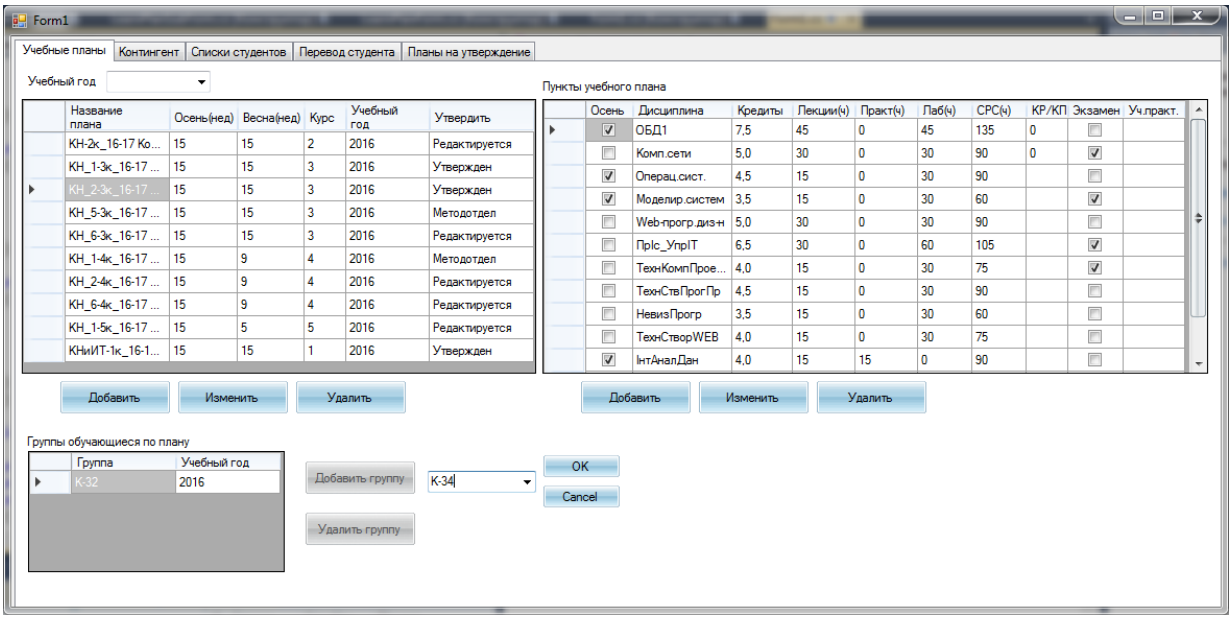

Рисунок 6.3 – Прихована панель для додавання нової групи

Для додавання нового навчального плану потрібно натиснути на кнопку «Добавить» під верхнім лівим табличним компонентом. При натисканні цієї кнопки з'являється форма для додавання навчального плану, як показано на рис. 6.4.

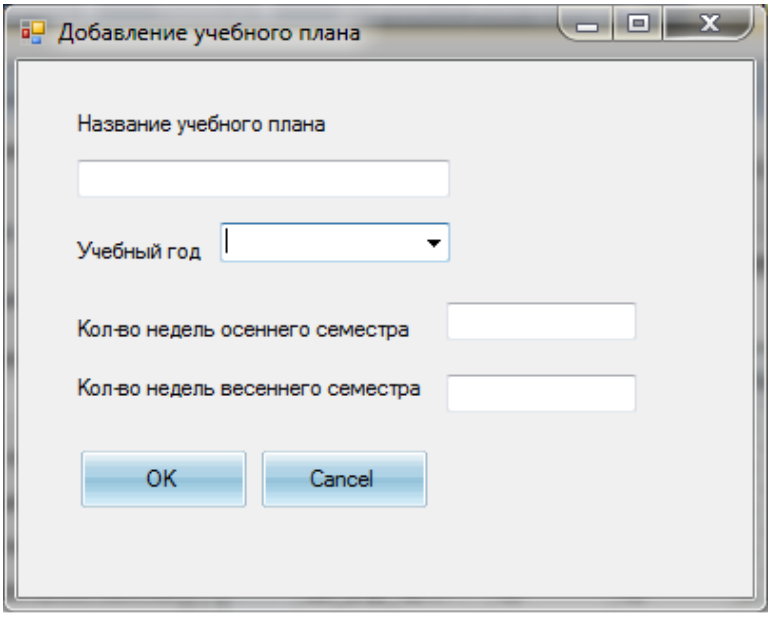

Рисунок 6.4 – Форма для додавання навчального плану

Користувач повинен заповнити поля і при натисканні на кнопку «ОК» додається навчальний план. Якщо навчальний план вже є, то користувач може натиснути кнопку «Изменить», щоб змінити обраний навчальний план, як показано на рис. 6.5.

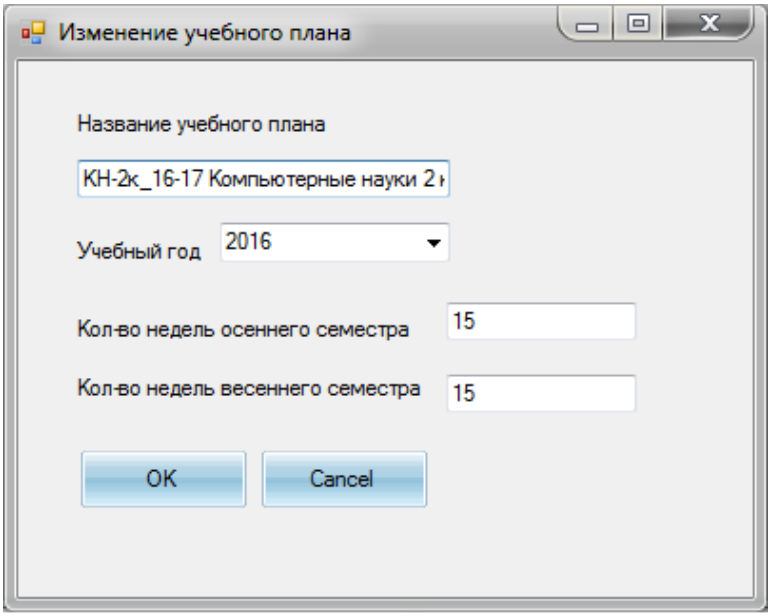

Рисунок 6.5 – Форма для зміни навчального плану

Якщо є навчальний план по начальному року, натиснувши на кнопку «Удалить» під верхнім лівим табличним компонентом, користувач може видалити навчальний план. Якщо з цього навчального плану у БД вже є інформація, то програма запитує підтвердження на видалення, що показано на рис. 6.6.

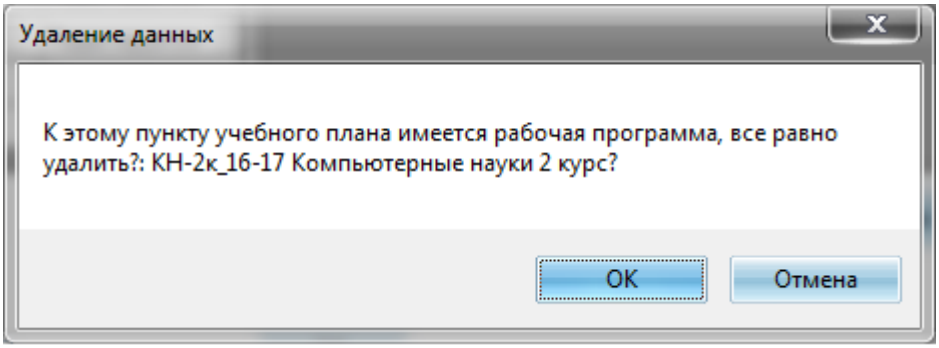

Рисунок 6.6 – Запит на видалення навчального плану

При натисканні кнопки «Добавить» під верхнім правим табличним компонентом для поточного навчального плану можна додати дисципліну.

З'являється форма для додавання дисципліни у перелік дисциплін даного навчального плану, як показано на рис. 6.7.

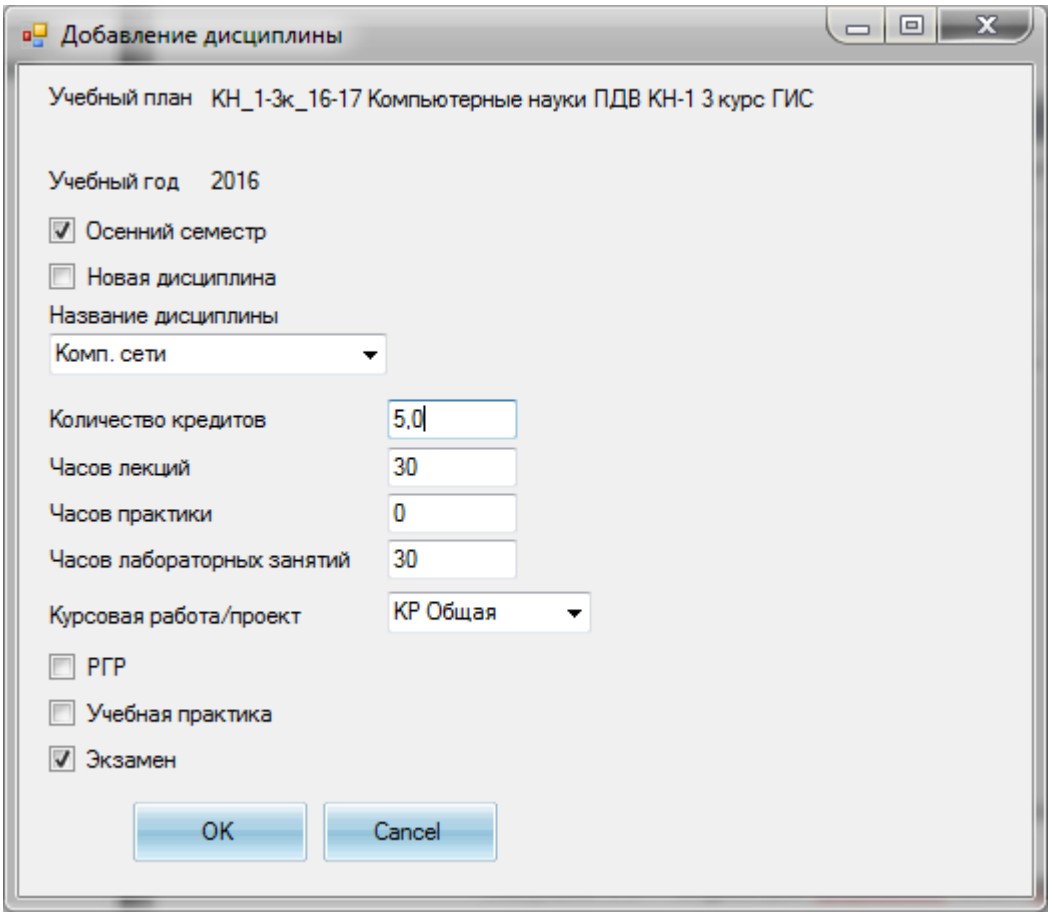

Рисунок 6.7 – Форма для додавання дисципліни у навчальний план

При натисканні пункту меню «Новая дисциплина» на формі стає доступною прихована панель для додавання нової дисципліни, якої ще немає у переліку навчальних дисциплін, що вивчаються в університеті. Для нової дисципліни потрібно задати повну та коротку назву, а також вибрати кафедру, викладачі якої будуть вести цю дисципліну.

Повна і коротка назва навчальної дисципліни повинні бути унікальними, інакше програма видасть повідомлення про те, що неможна додати у перелік дисциплін університету навчальну дисципліни з не унікальним повною або короткою назвою.

При натисканні пункту меню «Учебная практика» на формі стає можливим додавати к обраній дисципліні часи навчальної практики, як показано на рис. 6.8.

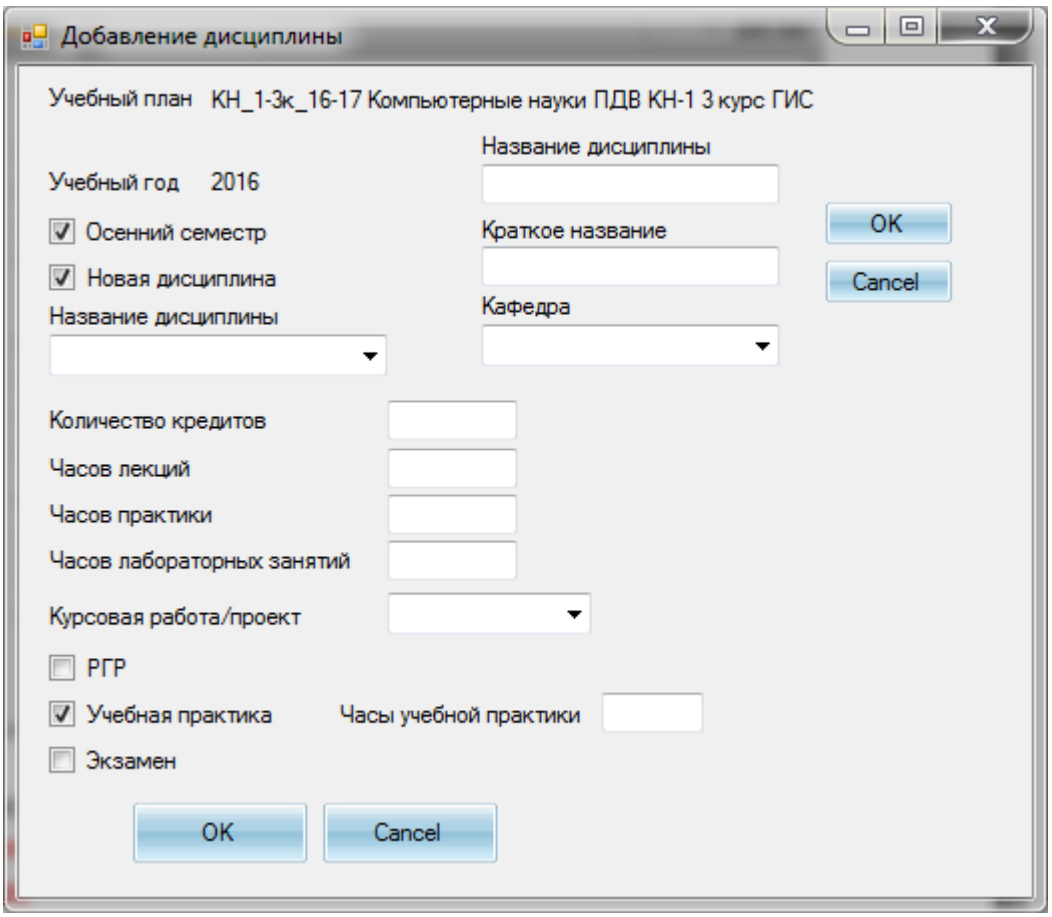

Рисунок 6.8 – Форма для додавання нової дисципліни з навчальною практикою

При натисканні на кнопку «Изменить» під верхнім лівим табличним компонентом відкривається форма для внесення змін у пункт навчального плану.

Користувач може змінити параметри існуючої дисципліни даного пункту навчального плану, або замінити цю дисципліну на іншу, якої ще немає у даному семестрі даного навчального плану.

Крім змінення власне дисципліни для пункту навчального плану, можна змінити кількість кредитів, призначених для неї, кількість годин занять різних видів: лекцій, практичних (семінарських), лабораторних, – наявність курсової роботи чи проекту, екзамену, розрахунково-графічної роботи, наявність та кількість годин навчальної практики як показано на рис. 6.9.

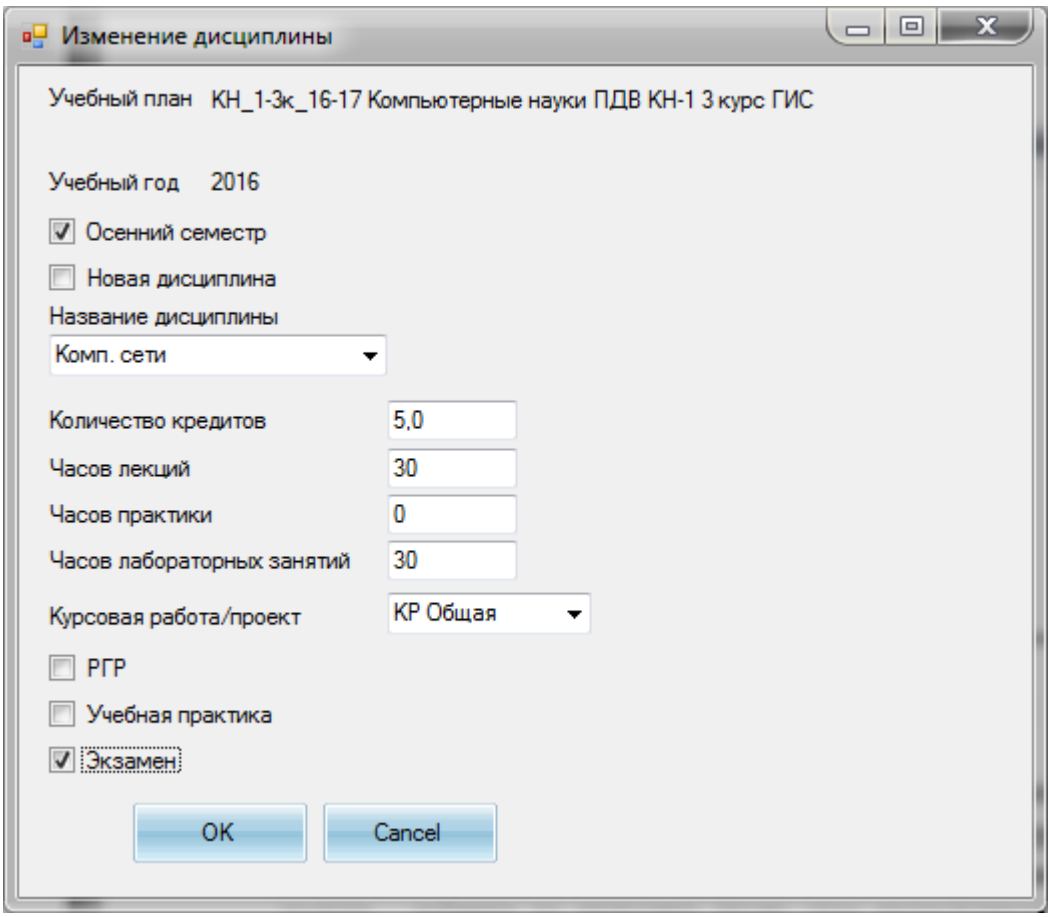

Рисунок 6.9 – Форма для зміни дисципліни в пункті навчального плану

Натиснувши на кнопку «Удалить» під верхнім лівим табличним компонентом користувач може видалити обрану дисципліну, програма запитує підтвердження на видалення, як показано на рис. 6.10.

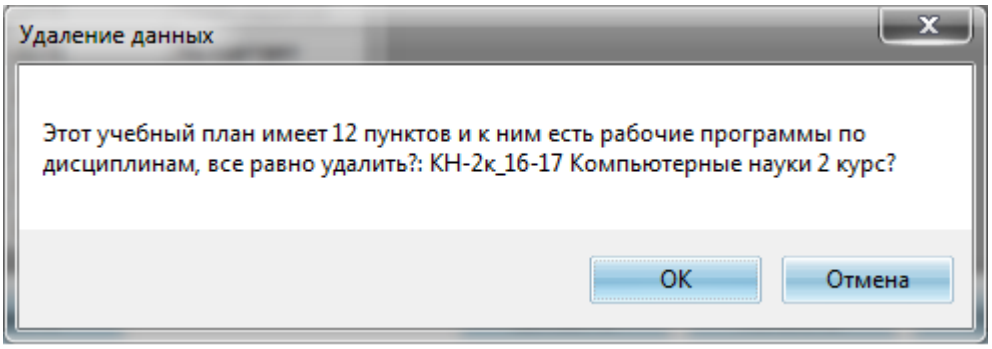

Рисунок 6.10 – Запит на видалення дисципліни

Для перегляду кількості студентів в академічних групах факультету потрібно перейти на вкладку «Контигент» (рис. 6.11).

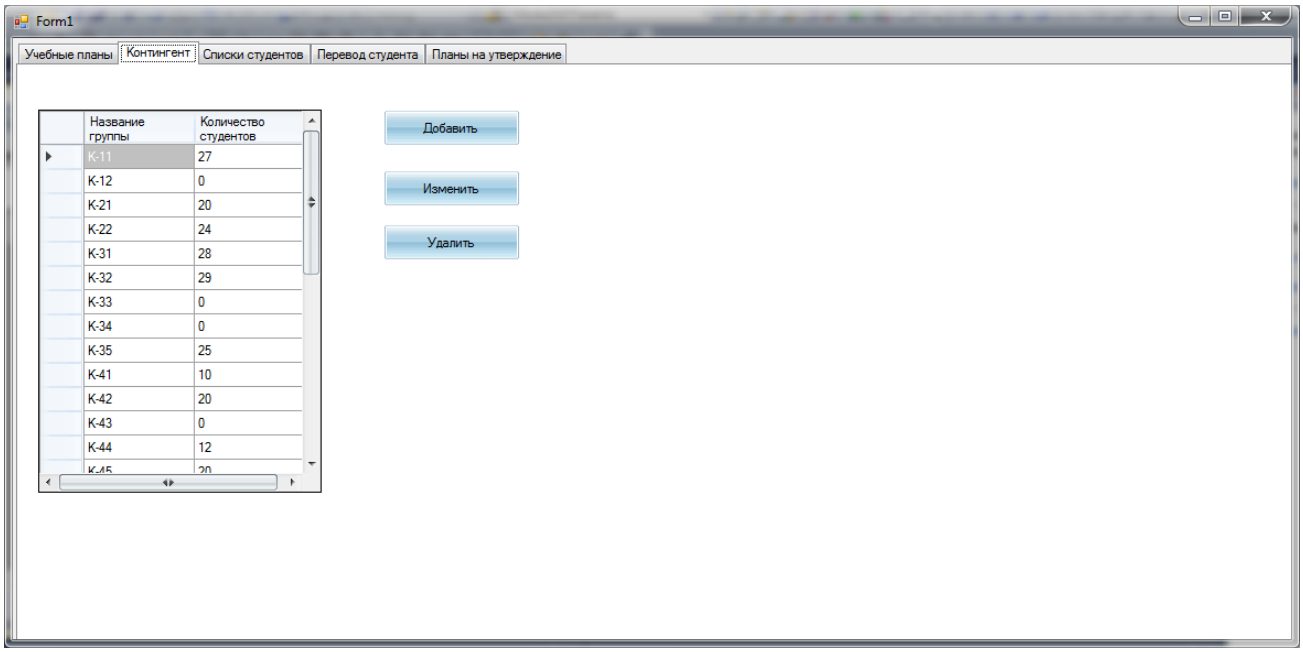

Рисунок 6.11 – Заповнена форма груп

Після натискання кнопки «Добавить», на головній формі стає доступною прихована панель для додавання нової групи та кількості студентів. Після того як на формі з'являється прихована панель кнопка додати , змінити та видалити групу стає недоступною, як показано на рис. 6.12.

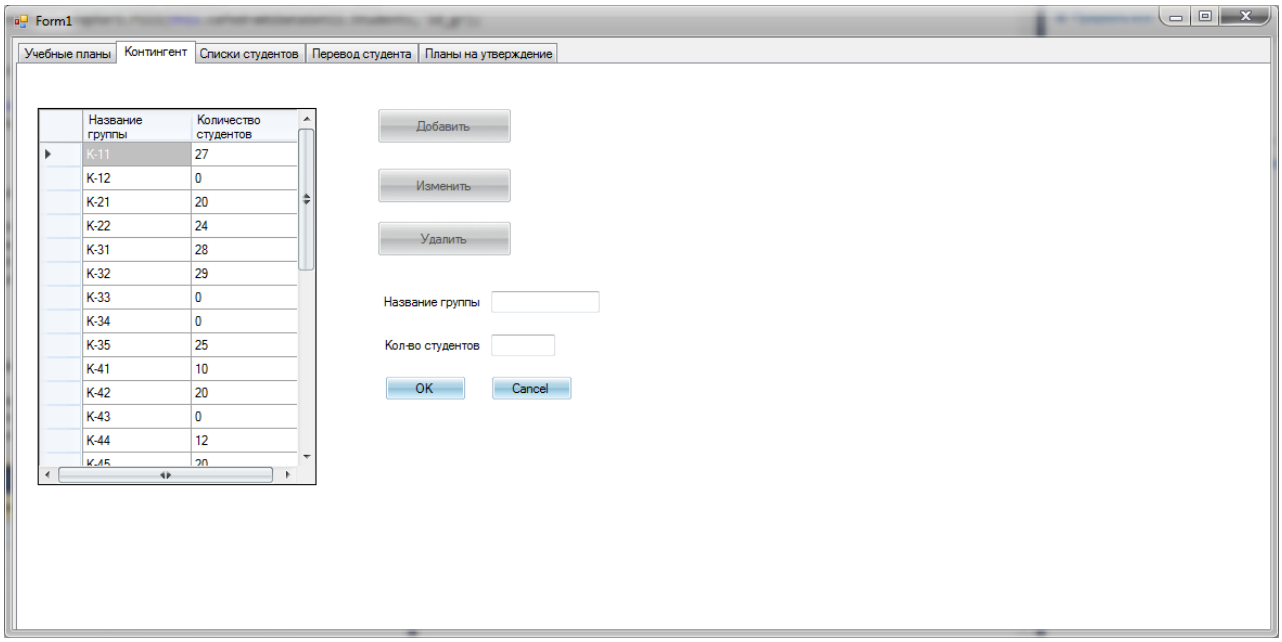

Рисунок 6.12 – Форма для зміни кількості студентів

Після вибору диспетчером деканату потрібної йому групи на екрані видна повна інформація про студентів (Прізвище, Ім'я, По-батькові, номер залікової книжки та рік прийому студента) (рис. 6.13).

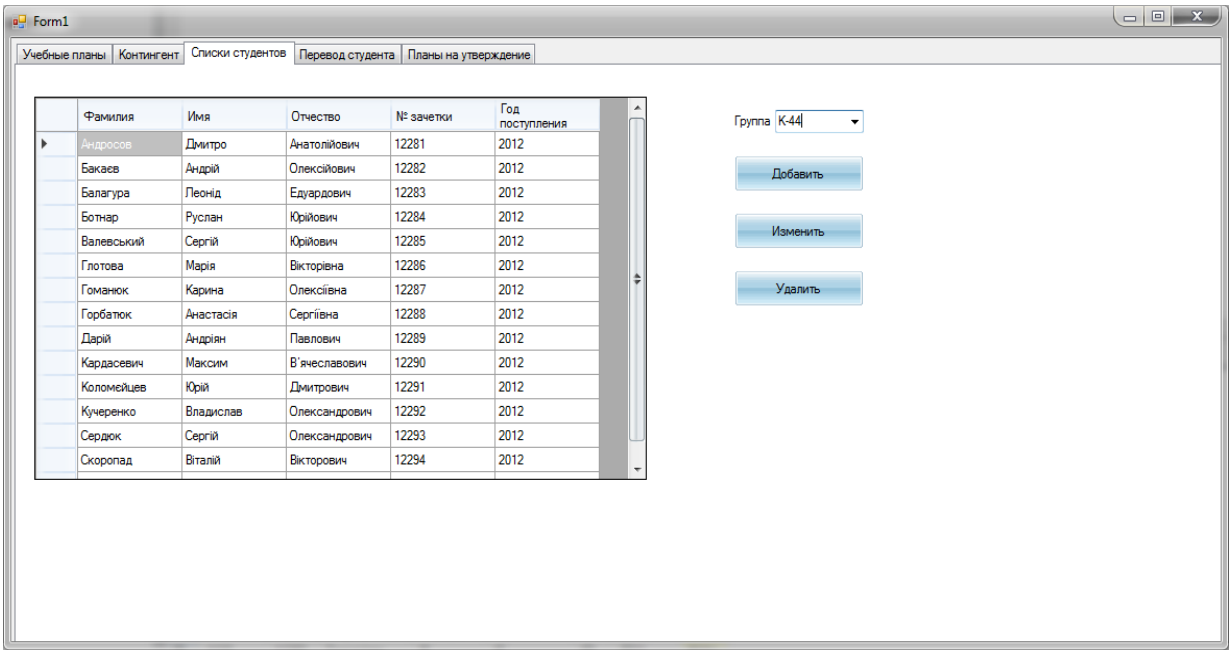

Рисунок 6.13 – Заповнена форма з інформацією про студенів

Для додавання нового студента потрібно натиснути на кнопку «Добавить». При натисканні цієї кнопки з'являється форма для додавання студента, як показано на рис. 6.14.

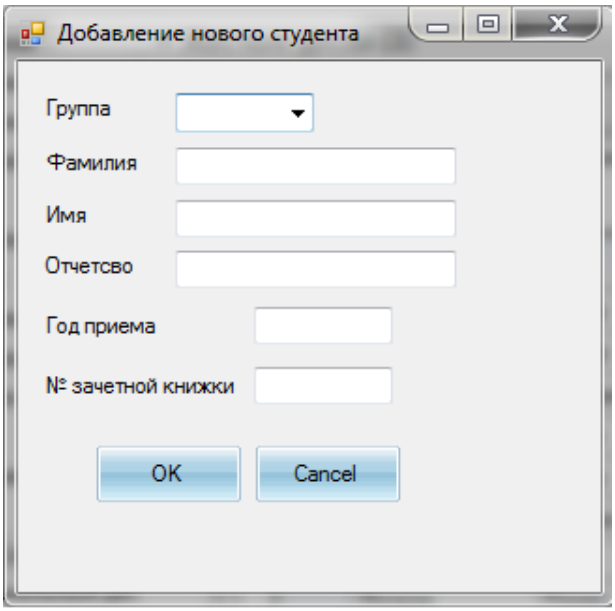

Рисунок 6.14 – Форма для додавання студента

Користувач повинен заповнити поля і при натисканні на кнопку «OK» додається новий студент. Якщо студент вже є, то диспетчер може натиснути кнопку «Изменить», щоб змінити обраного студента, як показано на рис. 6.15.

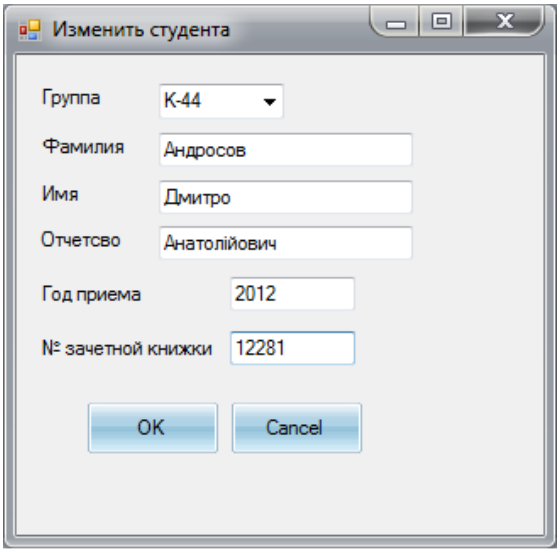

Рисунок 6.15 – Форма для зміни студента

На вкладці переведення студента, диспетчер деканату має можливість переводу всієї групи на наступний курс та перевод одного окремого студента із однієї групи у іншу, як показано на рис. 6.16.

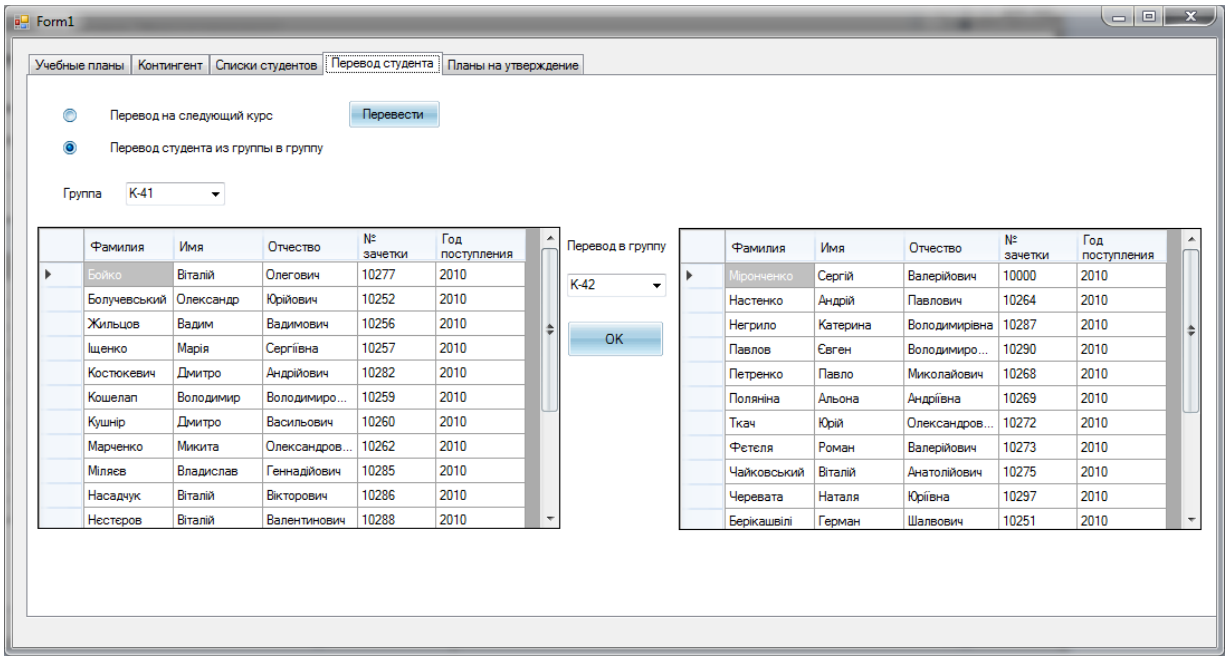

Рисунок 6.16 – Форма для переведення студента

На вкладці «Плани на затвердження» виводяться тільки ті плани які знаходяться у редагуванні (рисинок 6.17). Натиснувши на кнопку «Передать в методотдел» плани потрапляють на подальше редагування в методвідділ.

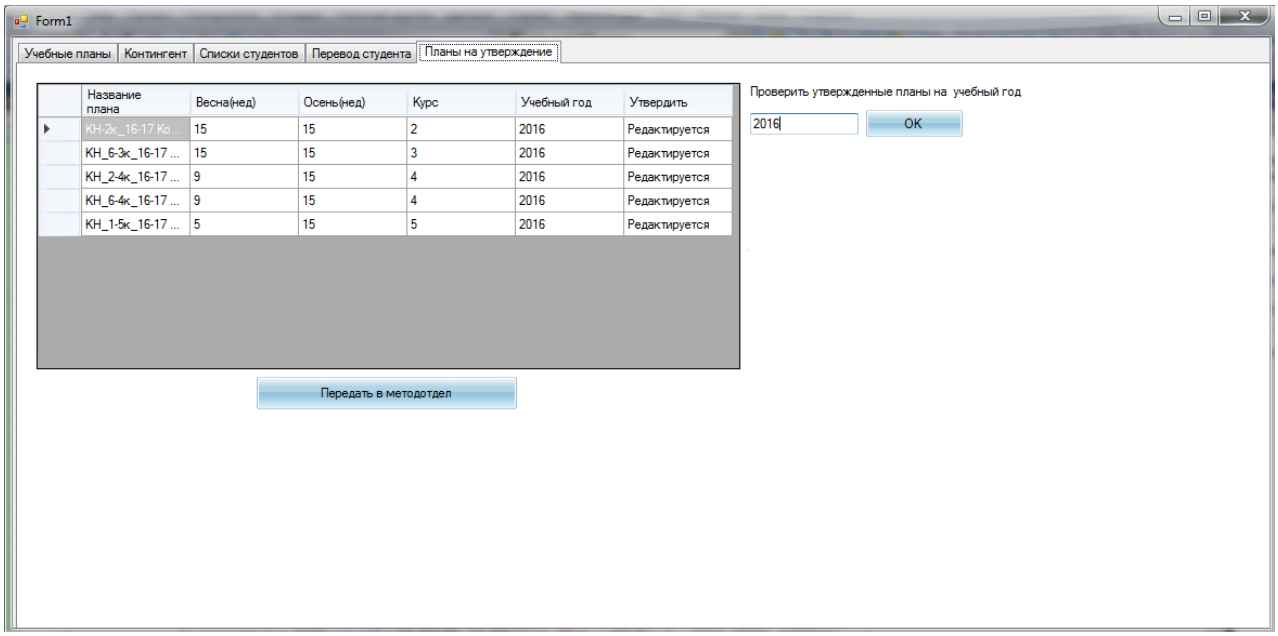

Рисунок 6.17 – Форма навчальних планів для методвідділу

При натисканні на кнопку «ОК», викликається процедура для розрахунку навчального навантаження викладачів згідно до дисциплін навчальних планів.

Спочатку процедура виконує перевірку наявності затверджених навчальних планів для усіх груп з ненульовим контингентом. Для всіх таких груп розраховується навчальне навантаження для кожної дисципліни по кожному виду навантаження. Якщо є групи з ненульовим контингентом без навчальних планів, або групи з нульовим контингентом з навчальними планами, виводиться відповідне повідомлення.

ВИСНОВКИ

Працівники деканату повинні постійно оновлювати навчальні плани, оновлювати інформацію по складу навчальних планів та інформацію про студентів.

Для звітності працівники деканату повинні впродовж семестру заповнювати навчальні плани, в яких записують кількість неділь у семестрі, курс та навчальний рік, заповнювати навчальний план новими дисциплінами чи оновлювати вже існуючі дисципліни. Оновлювати інформацію про студентів та групи.

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

Задачею даної частини комплексної магістерської роботи було було інтегрування у ІС «Університет» прикладних систем баз даних «Електронний деканат», «Інтегральні відомості», «Тестуюча система», а також створення програмного застосування у вигляді офісного додатку «АРМ співробітника деканату».

Особисто автором даної частини комплексної магістерської роботи були отримані результати:

- виконаний системний аналіз предметної області з метою оцінки повноти концептуальної моделі для вирішення поставленої задачі;
- доповнена концептуальна модель новими сутностями для підсхеми «Електронний деканат та поточна успішність студентів»;
- виконане логічне і фізичне проектування бази даних у відповідності до зміненої концептуальної моделі бази даних;
- розроблене програмне застосування «АРМ співробітника деканату»;

Розроблена інформаційна система «Університет» позволяє автоматизувати всі стадії навчального процесу в університеті. Для повноцінної роботи ІС в подальшому потрібно розробити програмного застосування для інших груп користувачів: співробітників методичного відділу, співробітників навчального відділу, викладачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Албахари Джозеф, Албахари Бен. C # 5.0. Справочник. Полное описание языка.: Пер. с англ. – Ю. Артеменко. – М.: Издательский дом «Вильямс», 2014. – 1008 с.
2. Дейт К. Дж.. Введение в системы баз данных, 6-е издание: Пер. с англ. – К.; М.; СПб: Издательский дом «Вильямс», 2000 – 848 с.
3. Джеми Макленнен, Чжаохуей Танг, Криват Богдан. Microsoft SQL Server 2008. Data Mining – интелектуальный анализ даннных.: Пер. с англ. – А. Лашкевич. – СПб.: БХВ-Петербург, 2009. – 720 с.
4. Душан Петкович. Microsoft SQL Server 2008. Руководство для начинающих. – СПб.: БХВ-Петербург, 2009. – 752 с.
5. Дэвидсон Л. Проектирование баз данных на SQL Server 2000 / Л.Дэвидсон; пер. с англ. – М. БИНОМ. Лаборатория знаний, 2003. – 680 с.
6. Ицик Бен-Ган. Microsoft SQL Server 2008. Основы T-SQL. – СПб.: БХВ-Петербург, 2009. – 430 с.
7. Майк Хотек. Microsoft SQL Server 2008. Реализация и обслуживание. Учебный курс Microsoft. – СПб. : Русская редакция, 2011. – 576 с.
8. Мамаев Е.В. Microsoft SQL Server 2000. – СПб.: БХВ-Петербург, 2004. – 1260 с.
9. Новиков Б.А. , Г. Р. Домбровская. Настройка приложений баз данных. – СПб.: БХВ-Петербург, 2006. – 729 с.
10. Никита Культин. Основы программирования в Microsoft Visual C# 2010. – СПб.: БХВ-Петербург, 2011. – 368 с.
11. Робин Дьюсон. SQL Server 2008 для начинающих разработчиков. – СПб.: БХВ-Петербург, 2009. – 704 с.
12. 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.
13. Эндрю Стиллмен, Дженнифер Грин. Изучаем C #. 3-е издание. – СПб.: Питер, 2014. – 816 с.

Д О Д А Т К И

ДОДАТОК А – ER-ДИАГРАММА ЛОГІЧНОЇ МОДЕЛІ БАЗИ ДАНИХ «УНІВЕРСИТЕТ»: ПІДСХЕМА «ЕЛЕКТРОННИЙ ДЕКАНАТ ТА ПОТОЧНА УСПІШНІСТЬ»

ДОДАТОК Б – ЛОГІЧНА СХЕМА БД: ПІДСХЕМА «ЕЛЕКТРОННИЙ ДЕКАНАТ ТА ПОТОЧНА УСПІШНІСТЬ»

ДОДАТОК В – ФІЗИЧНА СХЕМА БД: ТАБЛИЦІ ТА ПРЕДСТАВЛЕННЯ ДЛЯ ПІДСХЕМИ «ЕЛЕКТРОННИЙ ДЕКАНАТ ТА ПОТОЧНА УСПІШНІСТЬ»

ДОДАТОК Г – СЕРВЕРНЕ ПЗ: ЗБЕРЕЖЕНІ ПРОЦЕДУРИ ДЛЯ ПІДСХЕМИ «ЕЛЕКТРОННИЙ ДЕКАНАТ ТА ПОТОЧНА УСПІШНІСТЬ»

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 WindowsFormsApplication2
{
    public partial class Form1 : Form
    {

        private System.Data.SqlClient.SqlCommand DelLpCmd;
        private System.Data.SqlClient.SqlCommand DelLpSubCmd;
        byte id_f = 1;
        short id_gr, id_gr2;

        public Form1()
        {
            InitializeComponent();
            init_commands_DelLp();
            init_commands_DelLpSub();


            System.Data.SqlClient.SqlCommand GroupSelectCommand = new Sys-
tem.Data.SqlClient.SqlCommand("Select id_gr, name_gr from Groups where id_f =
"+id_f.ToString(), this.groupsTableAdapter.Connection);
            System.Data.ConnectionState previousConnectionState = GroupSelectCom-
mand.Connection.State;
            if (((GroupSelectCommand.Connection.State & System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open))
            {
                GroupSelectCommand.Connection.Open();
            }
            try
            {
                System.Data.SqlClient.SqlDataReader reader = GroupSelectCom-
mand.ExecuteReader();
                List_Item li;
                while (reader.Read())
                {
                    li = new List_Item(Convert.ToInt32(reader[0]), reader.GetString(1));
                    Gr1comboBox.Items.Add(li);
                }
                reader.Close();
            }
            finally
            {
                if ((previousConnectionState == System.Data.ConnectionState.Closed))
                    GroupSelectCommand.Connection.Close();
            }

            Gr1comboBox.SelectedIndex = -1;

        }

            private void init_commands_DelLp()
        {
            DelLpCmd = new System.Data.SqlClient.SqlCommand();
```

```
            DelLpCmd.Connection = this.learnPlanTableAdapter.Connection;
            DelLpCmd.CommandText = "LearnPlanDel";
            DelLpCmd.CommandType = System.Data.CommandType.StoredProcedure;
            DelLpCmd.Parameters.Add("@id_plan", System.Data.SqlDbType.SmallInt);
            DelLpCmd.Parameters.Add("@m", System.Data.SqlDbType.Bit);
            DelLpCmd.Parameters.Add("@k", System.Data.SqlDbType.Bit).Direction =
                                                            Sys-
tem.Data.ParameterDirection.InputOutput;
            DelLpCmd.Parameters.Add("@msg", System.Data.SqlDbType.VarChar, 100).Direction
=
                                                            Sys-
tem.Data.ParameterDirection.InputOutput;
        }

        private void init_commands_DelLpSub()
        {
            DelLpSubCmd = new System.Data.SqlClient.SqlCommand();
            DelLpSubCmd.Connection = this.viewLPSTableAdapter.Connection;
            DelLpSubCmd.CommandText = "LearnPlanSubDel";
            DelLpSubCmd.CommandType = System.Data.CommandType.StoredProcedure;
            DelLpSubCmd.Parameters.Add("@id_plan", System.Data.SqlDbType.SmallInt);
            DelLpSubCmd.Parameters.Add("@id_sub", System.Data.SqlDbType.SmallInt);
            DelLpSubCmd.Parameters.Add("@m", System.Data.SqlDbType.Bit);
            DelLpSubCmd.Parameters.Add("@k", System.Data.SqlDbType.Bit).Direction =
                                                            Sys-
tem.Data.ParameterDirection.InputOutput;
            DelLpSubCmd.Parameters.Add("@msg", System.Data.SqlDbType.VarChar,
100).Direction =
                                                            Sys-
tem.Data.ParameterDirection.InputOutput;
        }

        private void Form1_Load(object sender, EventArgs e)
        {

this.viewStudentsTableAdapter.Fill(this.cafedra01DataSet1.ViewStudents, id_gr2);
this.learnPlanTableAdapter.Fill(this.cafedra01DataSet1.LearnPlan);
this.studentsTableAdapter.Fill(this.cafedra01DataSet1.Students, id_gr);
this.groupsTableAdapter.Fill(this.cafedra01DataSet2.groups);
this.viewLPSTableAdapter.Fill(this.cafedra01DataSet.ViewLPS);
this.learnPlanViewTableAdapter.Fill(this.cafedra01DataSet.LearnPlanView);
this.viewGroupPlanNameGRTableAdapter.Fill(this.cafedra01DataSet1.ViewGroupPlanNameGR);

        }

        private void dataGridView1_RowEnter(object sender, DataGridViewCellEventArgs e)
        {

this.viewGroupPlanNameGRTableAdapter.FillBy(this.cafedra01DataSet1.ViewGroupPlanNameGR,
(short)(dataGridView1.Rows[e.RowIndex].Cells[9].Value));
            this.viewLPSTableAdapter.FillBy(this.cafedra01DataSet.ViewLPS,
(short)(dataGridView1.Rows[e.RowIndex].Cells[9].Value));

        }

        private void AddLpButt_Click(object sender, EventArgs e)
        {

            LearnPlanForm alp_params = new LearnPlanForm();
            alp_params.Show();
            alp_params.Hide();

            learnPlanParams alp_params = new learnPlanParams();
            alp_params.Header = "Добавление учебного плана";
            alp_params.add = true;
```

```
            alp_params.id_f = 0;
            alp_params.name_plan = "";
            alp_params.LearnYear = "";
            alp_params.weeks = "";
            alp_params.weeks2 = "";
            alp_params.z_f = false;
            alp_params.id_plan = 0;

            Form alpf = new LearnPlanForm(alp_params, learnPlanViewTableAdap-
ter.Connection);
            if (alpf.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            this.learnPlanViewTableAdapter.Fill(this.cafedra01DataSet.LearnPlanView);

        }

        private void EditLpButt_Click(object sender, EventArgs e)
        {
            LearnPlanParams elp_params = new learnPlanParams();
            elp_params.Header = "Изменение учебного плана";
            elp_params.add = false;
            elp_params.id_plan = (int)(dataGridView1.CurrentRow.Cells[9].Value);
            elp_params.name_plan = (string)(dataGridView1.CurrentRow.Cells[0].Value);
            elp_params.LearnYear = (string)(dataGridView1.CurrentRow.Cells[4].Value); ;
            elp_params.weeks = (int)(dataGridView1.CurrentRow.Cells[1].Value);
            elp_params.weeks2 = (int)(dataGridView1.CurrentRow.Cells[2].Value);
            Form elpf = new LearnPlanForm(elp_params, learnPlanViewTableAdap-
ter.Connection);
            if (elpf.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            this.learnPlanViewTableAdapter.Fill(this.cafedra01DataSet.LearnPlanView);

        }



        private void AddLpSubButt_Click(object sender, EventArgs e)
        {
            LearnPlanSubForm alps_params = new LearnPlanSubForm();
            alps_params.Show();
            alps_params.Hide();

            learnPlanSubParams alp_params = new learnPlanSubParams();
            alps_params.Header = "Добавление дисциплины";
            alps_params.add = true;
            alps_params.id_plan = 0;
            alps_params.id_sub = 0;
            alps_params.HourLect = "";
            alps_params.HourPract = "";
            alps_params.HourLab = "";

            Form alpsf = new LearnPlanSubForm(alps_params, viewLPSTableAdap-
ter.Connection);
            if (alpsf.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            this.viewLPSTableAdapter.Fill(this.cafedra01DataSet.ViewLPS);

         }
        private void EditLpSubButt_Click(object sender, EventArgs e)
        {
            LearnPlanSubParams elps_params = new learnPlanSubParams();
            elps_params.Header = "Изменение дисциплины";
            elps_params.add = false;
            elps_params.id_plan = (int)(dataGridView1.CurrentRow.Cells[9].Value);
            elps_params.id_sub = (int)(dataGridView1.CurrentRow.Cells[8].Value);
            elps_params.HourLect = (int)(dataGridView1.CurrentRow.Cells[1].Value);
            elps_params.HourPract = (int)(dataGridView1.CurrentRow.Cells[2].Value);
            elps_params.HourLab = (int)(dataGridView1.CurrentRow.Cells[3].Value);
```

```
            Form elpf = new LearnPlanForm(elps_params, viewLPSTableAdapter.Connection);
            if (elpf.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            this.viewLPSTableAdapter.Fill(this.cafedra01DataSet.ViewLPS);


        }


        private void AddStudButt_Click(object sender, EventArgs e)
        {

            StudForm as_params = new StudForm();
            as_params.Show();
            as_params.Hide();

            StudParams as_params = new StudParams();
            as_params.Header = "Добавление нового студента";
            as_params.add = true;
            as_params.surname = "";
            as_params.name1 = "";
            as_params.name2 = "";
            as_params.number = "";
            as_params.id_gr = 0;
            as_params.year_enter = "";
            as_params.id_student = 0;

            Form alpf = new LearnPlanSubForm(as_params, viewLPSTableAdapter.Connection);
            if (alpf.ShowDialog() == System.Windows.Forms.DialogResult.OK)
            this.viewLPSTableAdapter.Fill(this.cafedra01DataSet.ViewLPS);


        }

        private void Gr1comboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_gr = (short)(((List_Item)(Gr1comboBox.SelectedItem)).id);

            this.studentsTableAdapter.Fill(this.cafedra01DataSet1.Students, id_gr);

        }

        private void Gr2comboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_gr = (short)(((List_Item)(Gr2comboBox.SelectedItem)).id);

            this.studentsTableAdapter.Fill(this.cafedra01DataSet1.Students, id_gr);


        }

      private void Gr3comboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_gr2 = (short)(((List_Item)(Gr3comboBox.SelectedItem)).id);

            this.viewStudentsTableAdapter.Fill(this.cafedra01DataSet1.ViewStudents,
id_gr2);

        }

        private void DelLpButt_Click_1(object sender, EventArgs e)
        {
            if (MessageBox.Show("Этот учебный план имеет 12 пунктов и к ним есть рабочие
программы по дисциплинам, все равно удалить?: " +
(string)(dataGridView1.CurrentRow.Cells[1].Value) + "?",
```

```
                "Удаление данных ", MessageBoxButtons.OKCancel) == Sys-
tem.Windows.Forms.DialogResult.OK)
            {
                DelLpCmd.Parameters["@id_plan"].Value =
(int)(dataGridView1.CurrentRow.Cells[9].Value);
                DelLpCmd.Parameters["@m"].Value = 0;
                DelLpCmd.Parameters["@k"].Value = System.DBNull.Value;
                DelLpCmd.Parameters["@msg"].Value = System.DBNull.Value;
                System.Data.ConnectionState previousConnectionState =
DelLpCmd.Connection.State;
                if (((DelLpCmd.Connection.State & System.Data.ConnectionState.Open) !=
System.Data.ConnectionState.Open))
                {
                    DelLpCmd.Connection.Open();
                }
                try
                {
                    DelLpCmd.ExecuteNonQuery();
                    if ((short)(DelLpCmd.Parameters["@k"].Value) > 0)
                    {
                        if (MessageBox.Show((string)(DelLpCmd.Parameters["@msg"].Value) +
"\nВсе равно удалить?",
                            "Удаление из связанных таблиц", MessageBoxButtons.OKCancel)
== System.Windows.Forms.DialogResult.OK)
                        {
                            DelLpCmd.Parameters["@m"].Value = 1;
                            DelLpCmd.ExecuteNonQuery();
                        }
                    }
                }
                finally
                {
                    if ((previousConnectionState == System.Data.ConnectionState.Closed))
                        DelLpCmd.Connection.Close();
                }
                this.learnPlanViewTableAdapter.Fill(this.cafedra01DataSet.LearnPlanView);
        }

    }
          private void DelLpSubButt_Click(object sender, EventArgs e)
        {
         if (MessageBox.Show("К этому пункту учебного плана имеется рабочая программа,
все равно удалить?: " + (string)(dataGridView2.CurrentRow.Cells[13].Value) + "?",
                "Удаление данных ", MessageBoxButtons.OKCancel) == Sys-
tem.Windows.Forms.DialogResult.OK)
            {
                DelLpSubCmd.Parameters["@id_plan"].Value =
(int)(dataGridView1.CurrentRow.Cells[9].Value);
                DelLpSubCmd.Parameters["@m"].Value = 0;
                DelLpSubCmd.Parameters["@k"].Value = System.DBNull.Value;
                DelLpSubCmd.Parameters["@msg"].Value = System.DBNull.Value;
                System.Data.ConnectionState previousConnectionState =
DelLpSubCmd.Connection.State;
                if (((DelLpSubCmd.Connection.State & System.Data.ConnectionState.Open) !=
System.Data.ConnectionState.Open))
                {
                    DelLpSubCmd.Connection.Open();
                }
                try
                {
                    DelLpSubCmd.ExecuteNonQuery();
                    if ((short)(DelLpSubCmd.Parameters["@k"].Value) > 0)
                    {
                        if
(MessageBox.Show((string)(DelLpSubCmd.Parameters["@msg"].Value) + "\nВсе равно удалить?",
```

```
                            "Удаление из связанных таблиц", MessageBoxButtons.OKCancel)
== System.Windows.Forms.DialogResult.OK)
                        {
                            DelLpSubCmd.Parameters["@m"].Value = 1;
                            DelLpSubCmd.ExecuteNonQuery();
                        }
                    }
                }
                finally
                {
                    if ((previousConnectionState == System.Data.ConnectionState.Closed))
                        DelLpSubCmd.Connection.Close();
                }
                this.viewLPSTableAdapter.Fill(this.cafedra01DataSet.ViewLPS);
            }
        }
    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;
        }
    }
}
```