МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської та аспірантської підготовки

Кафедра інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

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

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

Зубенко Сергій Сергійович

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

Консультант

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

Одеса 2018

АНОТАЦІЯ

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

Наукова новизна роботи полягає в тому, що у даний роботі розглядається весь навчальний процес цілком, з урахуванням взаємодії та зв'язків окремих складових частин цього процесу. Дана робота розглядає складову частину навчального процесу, яка відноситься до роботи працівника навчального відділу закладу вищої освіти.

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

Об'єктом дослідження є навчальний процес у закладі вищої освіти.

Предметом дослідження є засоби автоматизації даного процесу.

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

Результати даної роботи можуть використовуватись для автоматизації роботи працівника навчального відділу при складанні розкладу занять та проведенні навчального процесу у будь-якому закладі вищої освіти.

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

Обсяг роботи 62 сторінки. Вона містить 30 рисунків та 15 посилань.

SUMMARY

The master's work on the topic “Modeling and development of the subsystem “Educational department” as part of the information system “Educational process of the university”” is caused by the task of automating the main works that provide the educational process at the university. This determines the relevance of this work.

The scientific novelty of the work lies in the fact that in the present work the entire educational process is considered in its entirety, taking into account the interactions and connections of the individual components of this process. This paper considers an integral part of the educational process, which relates to the work of an employee of the educational department of a higher education institution.

The purpose and objective of the study is to study the methods and models of work automation that ensure the conduct of the educational process in a higher education institution.

The object of study is the educational process in institutions of higher education.

The subject of research is the automation of this process.

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

The results of this work can be used to automate the work of an employee of the educational department when drawing up a schedule of classes and conducting an educational process in any institution of higher education.

Keywords: INFORMATION SYSTEM, RELATIONAL DATABASE, IDEF0 CONTEXT DIAGRAM, ER-DIAGRAM, SERVER SOFTWARE, AUTOMATED WORKPLACE, CLASS SCHEDULE.

The amount of work 62 pages. It contains 30 figures and 15 references.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....7
ВСТУП 8
2 МЕТОДОЛОГІЯ ФУНКЦІОНАЛЬНОГО МОДЕЛЮВАННЯ (IDEF0).....10
2.1 Розробка контекстної діаграми IDEF0 12
2.1 Декомпозиція контекстної діаграми IDEF0 13
3 СИСТЕМНИЙ АНАЛІЗ І КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ.....16
3.1 Етапи проектування бази даних 16
3.2 Концептуальне проектування бази даних 17
3.2.1 Визначення основних типів сутностей та типів зв’язків 17
3.2.2 Визначення атрибутів і зв’язування їх з типами сутностей і зв’язків 25
3.2.3 Визначення атрибутів, що є потенційними і первинними ключами27
3.2.4 Визначення відповідності концептуальної моделі транзакціям користувачів.....27
4 ВИБІР СУБД ТА СЕРЕДОВИЩА РОЗРОБКИ ЗАСТОСУВАННЯ.....32
4.1 Вибір СУБД.....32
4.2 Вибір середовища розробки застосування та мови програмування 32
5 ЛОГІЧНЕ І ФІЗИЧНЕ ПРОЕКТУВАННЯ БД 35
5.1 Логічне проектування.....35
5.2 Фізичне проектування 36
6 СЕРВЕРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ.....38
7 ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ 43
7.1 Загальні відомості.....43
7.2 Функціональне призначення 43
7.3 Керівництво програміста 43
7.4 Посібник користувача 48
ВИСНОВКИ.....62
ПЕРЕЛІК ПОСИЛАНЬ.....64
Д О Д А Т К И ДОДАТОК А Логічна схема бази даних 65
ДОДАТОК А Логічна схема бази даних.....66
ДОДАТОК Б Фізична схема бази даних.....67
ДОДАТОК В Серверне програмне забезпечення 69
ДОДАТОК Г Вихідний код клієнтського застосування 69

ПЕРЕЛІК СКОРОЧЕНЬ

АРМ – автоматизоване робоче місце
БД – база даних
ЗВО – заклад вищої освіти
ІС – інформаційна система
МОН – міністерство освіти та науки
ОНП – освітньо-наукова програма підготовки фахівців
ОПП – освітньо-професійна програма підготовки фахівців
СУБД – система управління базами даних

ВСТУП

Навчальний відділ університету повинен забезпечувати виконання основної задачі закладу вищої освіти – надання студентам можливості отримання вищого рівня освіти. Працівники навчального відділу розробляють розклад занять, за яким навчаються студенти, контролюють проведення цих занять, перевіряють виконання викладачами інших видів діяльності, пов’язаних з наданням студентам освітніх послуг.

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

З переліку вихідних даних, необхідних для автоматизованого складання розкладу занять зрозуміло, що недоцільно розробляти окрему інформаційну систему (ІС), спрямовану лише для вирішення цієї приватної задачі, оскільки доведеться вводити великі обсяги даних, потрібних для складання розкладу занять.

Але якщо розглядати задачу автоматизованого складання розкладу занять у контексті єдиної інформаційної системи «Навчальний процес університету», то кількість даних, потрібних лише для процесу розкладу занять радикально зменшується. Інформаційна система, яка працює з усіма складовими навчального процесу університету, дозволяє користувачам оперувати тільки тими даними, що відносяться до їх компетенції.

Таким чином, основні таблиці-довідники, наприклад, перелік підрозділів університету, вносить у базу даних адміністратор ІС. Список викладацького составу університету, вносять декілька користувачів групи «Працівник кафедри», які відповідають за дані у межах своєї кафедри. Навчальні плани у базу даних (БД) інформаційної системи вносять відповідальні працівники деканатів факультетів, вони ж вносять у БД список академічних груп факультету та їх контингент.

При проектуванні вказаної єдиної інформаційної системи виявляється, що вона повинна містити багато складових частин, також виявляється декілька груп користувачів, які будуть працювати з цією системою. Тому після виявлення основних змістовних блоків діаграми декомпозиції IDEF0 та

основних груп користувачів потрібно розділити процес проектування ІС на моделювання та проектування декількох підсистем у складі єдиної ІС. Всі підсистеми повинні мати спільні таблиці-довідники, що містять основні дані по навчальному процесу університету.

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

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

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

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

1.1 Об’єкти та суб’єкти навчального процесу університету

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

Освіту студент набуває за допомогою аудиторних занять та самостійної роботи. Для можливості проведення аудиторних занять навчальний відділ університету складає розклад занять на кожний поточний семестр. Основою для складання розкладу занять є навчальні плани за спеціальностями, контингент студентів, та заплановане аудиторне навантаження викладачів.

Робочі навчальні плани розроблять деканати факультетів на основі освітньо-професійних програм (ОПП) та освітньо-наукових програм (ОНП) зі спеціальності. Розроблені навчальні плани затверджуються методичним відділом університету.

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

На кафедрах відбувається розподіл навчального навантаження між викладачами. Навчальне навантаження викладача складається з аудиторних занять, керівництва курсовим та дипломним проектуванням студентів,

проведення та перевірки результатів контролюючих заходів. Викладачі кафедри повинні планувати навчальне навантаження на наступний рік у обсязі, що обумовлений нормативними документами МОН. Також наприкінці навчального року викладачі повинні звітувати про виконання навчального навантаження за рік.

Частина навчального навантаження викладача – аудиторне навантаження – є одною з складових частин для складання розкладу занять у ЗВО. Іншими складовими частинами є робочі навчальні плани, контингент студентів, аудиторний фонд ЗВО, вимоги викладачів до власного розкладу занять та нормативи ЗВО до якісного розкладу занять студентів.

1.2 Задачі працівників навчального відділу університету

Навчальний відділ розробляє розклад занять у вищі. Також працівники цього відділу контролюють проведення аудиторних занять протягом року.

Наприкінці навчального року навчальний відділ перевіряє розрахунок запланованого навчального навантаження кафедр університету на наступний навчальний рік та розподіл цього навантаження між викладачами кафедри.

На початку навчального року навчальний відділ перевіряє звітність кафедр щодо виконання навчального навантаження викладачами кафедри у минулому навчальному році.

Інформаційна система «Навчальний процес університету» дозволить спростити потік документів між підрозділами університету. Включення до ІС програми автоматизованого складання розкладу занять значно спростить роботу працівників навчального відділу по розробці розкладу занять та адаптуванні його для викладачів університету.

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

2 МЕТОДОЛОГІЯ ФУНКЦІОНАЛЬНОГО МОДЕЛЮВАННЯ (IDEF0)

2.1 Розробка контекстної діаграми IDEF0

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

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

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

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

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

Перелік запитань:

- Які навчальні плани повинні бути у наступному навчальному році?
- Які академічні групи навчаються за яким навчальним планом?
- Які лекції читаються для потоку академічних груп?
- Які викладачі проводять заняття по дисциплінах навчального плану?
- В яких аудиторіях повинні проводитись конкретні заняття академічних груп?
- Як враховуються вимоги викладачів до розкладу занять?
- В яких групах по яких дисциплінах проводяться іспити?
- Яке навчальне навантаження заплановане викладачам кафедр на наступний навчальний рік?
- Як викладачі виконали навчальне навантаження за минулий рік?
- Чи наявні всі дані для складання розкладу занять?

Контекстна діаграма IDEF0 навчального процесу ЗВО зображена на рис. 2.1.

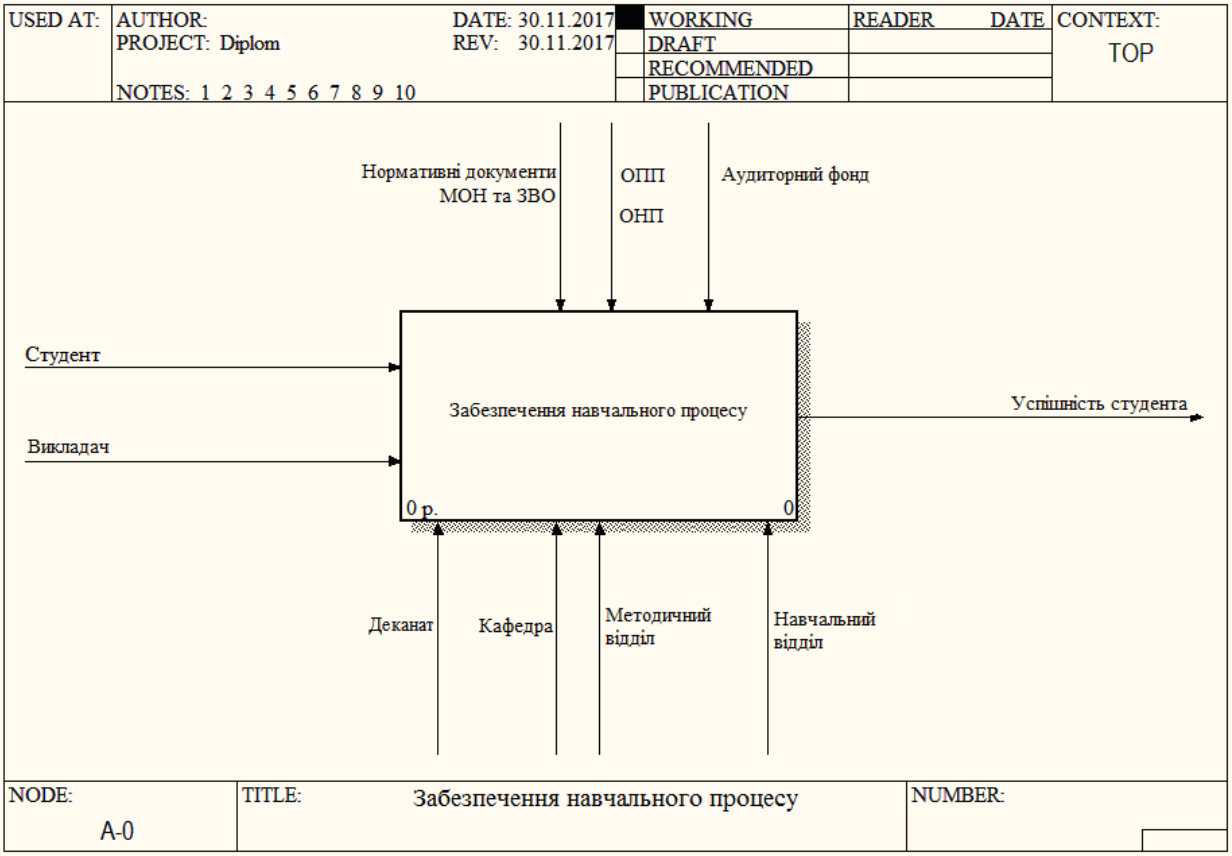

Рисунок 2.1 – Контекстна діаграма IDEF0

У контекстній діаграмі присутні такі граничні стрілки:

1) Входи: студент, викладач.

2) Управління: нормативні документи МОН та ЗВО, ООП, ОНП, аудиторний фонд.

3) Механізми: деканат, методичний відділ, кафедра, навчальний відділ.

4) Виходи: успішність студента.

2.1 Декомпозиція контекстної діаграми IDEF0

Відповідно до стандарту, на кожному рівні декомпозиції IDEF0 повинен використовуватися принцип обмеження об'єкта, тому відповідно до цього принципу вважається, що єдиний блок і кілька стрілок на самому верхньому (контекстному) рівні використовуються для визначення кордону всієї системи. Відповідно, стрілки, що стосуються цього блоку, описують головні управління, входи, виходи і механізми цієї системи.

З опису навчального процесу ЗВО зрозуміло, що декомпозиція контекстної діаграми IDEF0 повинна включати блоки розробки робочих

навчальних планів, розподілу навчального навантаження між викладачами, складання розкладу занять, проведення поточного навчального процесу.

На рис. 2.1 зображено, як в загалі працює система «Навчальний процес університету». На рисунку 2.2 діаграма декомпозиції IDEF0 більш ретельно представляє саме процес складання розкладу занять, який є основним процесом у діяльності навчального відділу.

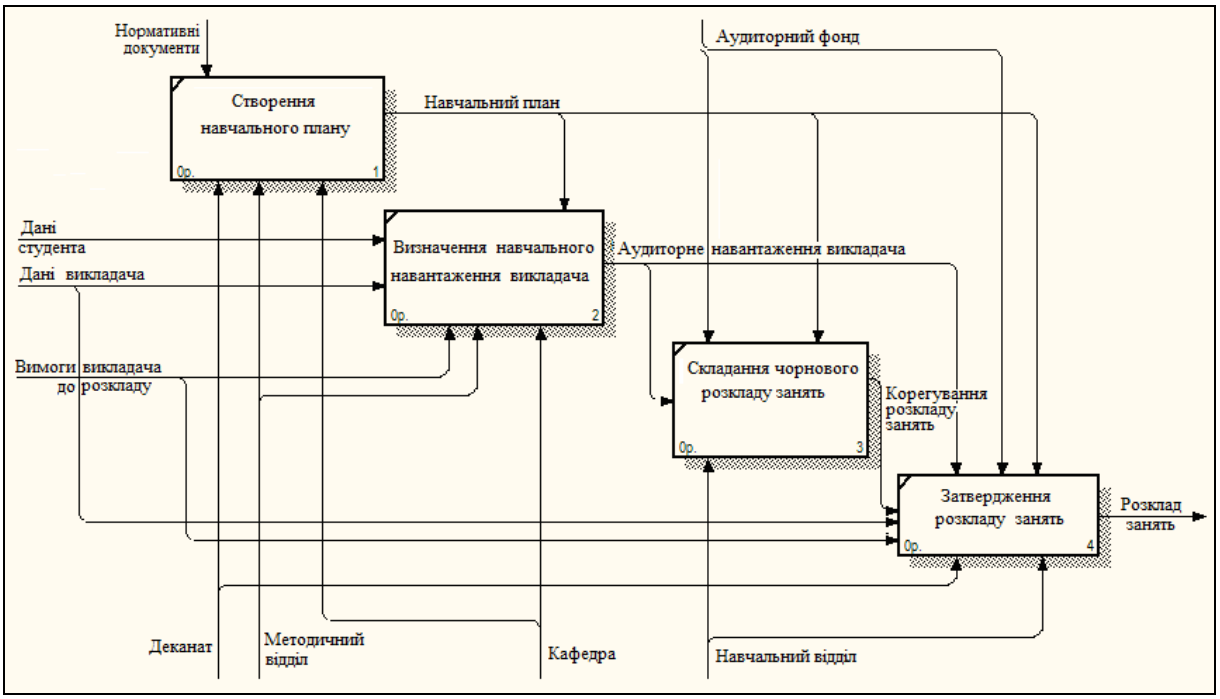

Рисунок 2.2 – Діаграма декомпозиції IDEF0

Навчальний відділ університету повинен забезпечувати виконання основної задачі закладу вищої освіти – надання студентам можливості отримання вищого рівня освіти. Працівники навчального відділу розробляють розклад занять, за яким навчаються студенти, контролюють проведення цих занять, перевіряють виконання викладачами інших видів діяльності, пов’язаних з наданням студентам освітніх послуг.

Розклад занять на наступний семестр складається згідно робочим навчальним планам, за якими навчаються академічні групи університету. Для можливості складання розкладу занять навчальний відділ отримує від кафедр університету навчальне навантаження викладачів, тобто розподіл аудиторних занять між викладачами кафедри.

Деканати факультетів надають до навчального відділу контингент студентів – кількісний склад всіх академічних груп, що будуть навчатись у наступному семестрі.

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

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

3 СИСТЕМНИЙ АНАЛІЗ І КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ

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

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

3.1 Етапи проектування бази даних

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

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

Основні етапи концептуального проектування [3]:

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

Завданням логічного проектування бази даних є створення логічної моделі на основі обраної моделі даних, але без урахування конкретної СУБД,

яка буде використовуватися, та інших фізичних аспектів реалізації інформаційної системи. На етапі логічного проектування отримана концептуальна модель уточнюється і перетвориться для відповідності структурам даних і зв'язків між ними, притаманних обраної моделі даних: реляційної, об'єктно-орієнтованої, об'єктно-реляційної, тощо.

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

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

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

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

3.2 Концептуальне проектування бази даних

3.2.1 Визначення основних типів сутностей та типів зв’язків

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

Основними групами користувачів будуть:

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

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

Для вказаної групи користувачів основними транзакціями будуть:

1) Отримати від факультетів робочі навчальні плани на наступний навчальний рік.
2) Отримати від факультетів контингенти студентів на наступний семестр (списки академічних груп з запланованою кількістю студентів в них).
3) Отримати від факультетів прикріплення академічних груп до спеціальностей та освітньо-професійних програм (ООП) навчання.
4) Отримати від методичного відділу списки дисциплін, закріплених за кафедрами на наступний навчальний рік.
5) Отримати від кафедр список потокових лекцій по дисциплінах кафедри.
6) Отримати від кафедр список лабораторних занять по дисциплінах кафедр, які проводяться з поділом груп на підгрупи.
7) Отримати від кафедр розподіл між викладачами кафедри аудиторних занять по дисциплінах кафедри на наступний семестр.
8) Відновити дані по аудиторному фонду університету на наступний семестр.
9) Отримати від кафедр перелік побажань та вимог викладачів до розкладу занять на наступний семестр.
10) Скласти чорновий розклад занять на наступний семестр.
11) Передати чорновий розклад занять на кафедри для можливого корегування розкладу.
12) Отримати від кафедр дані для корегування розкладу занять.
13) Скласти остаточний розклад занять на наступний семестр.

14) Отримати від кафедр повне заплановане навчальне навантаження викладачів на наступний навчальний рік та затвердити його.
15) Перевірити виконання викладачами навчального навантаження у попередньому навчальному році.

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

- «Факультет»;
- «Кафедра»;
- «Навчальний план»;
- «Група»;
- «Викладач»;
- «Дисципліна (навчальна)»;
- «Вид занять»;
- «Заняття»;
- «Аудиторія»;
- «Потік (лекційний)».

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

Факультет має декілька академічних груп та декілька навчальних планів – між типами сутностей «Факультет» та «Навчальний план» існує тип зв’язку «один до багатьох»; так саме, як і між типами сутностей «Факультет» та «Група».

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

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

Деякі навчальні дисципліни можуть зустрічатись у декількох навчальних планах, наприклад, загальноосвітні дисципліни. У різних навчальних планах для таких дисциплін може передбачатись різне аудиторне навантаження, тобто різна кількість занять різного виду.

Тому необхідно у схему бази даних додати похідний тип сутності «Дисципліна плану» (пункт навчального плану, дисципліна навчального плану), відповідно до якого і потрібно визначати заняття з деякої дисципліни для деякої групи.

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

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

Усі три види занять зустрічаються для однієї дисципліни досить рідко, зазвичай планується проведення лекцій та одного з видів практичних занять: семінари або лабораторні роботи. Інколи для дисципліни можуть бути призначені тільки лекції, або тільки практичні заняття.

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

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

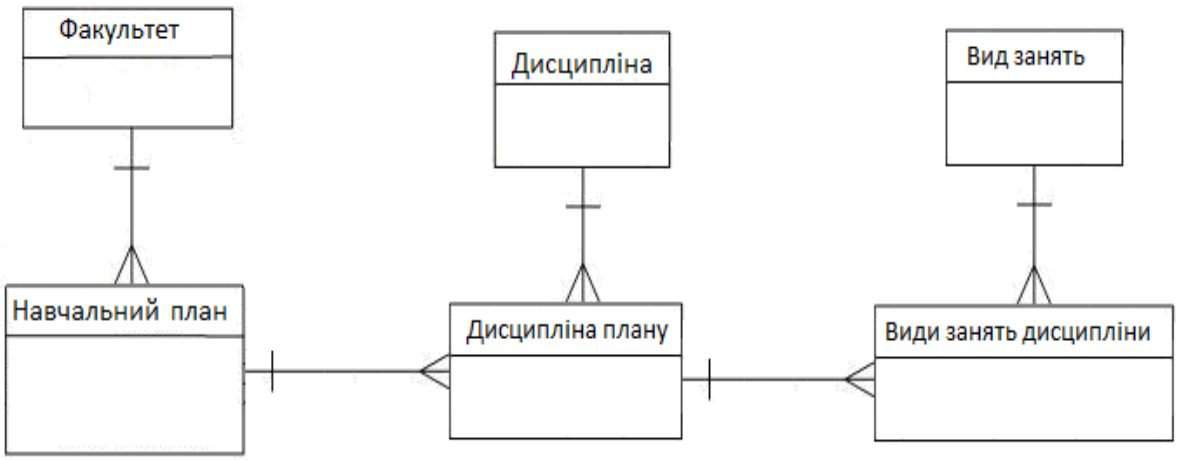

Рисунок 3.1 – Перша концептуальна модель БД підсхеми «Розклад занять»

Навчальний план зазвичай складають на весь навчальний рік, тобто на обидва семестри. Але розклад занять для двох семестрів складається окремо та у різний час. Можна у для типу сутностей «Дисципліна плану» ввести атрибут, що буде позначати, у якому семестрі – восени або весною – вивчається дана дисципліна. Але це не дуже зручний підхід.

Тим більше такий підхід не зручний, що у різних семестрах може бути різна кількість навчальних тижнів, які використовуються для розрахунку кількості годин занять групи з кожної дисципліни на тиждень. Тому краще позначку семестру ввести у тип сутності «Навчальний план», тобто для одного навчального року у базі даних буде зберігатись два екземпляри сутності «Навчальний план» для кожного курсу – якщо для цього курсу цієї спеціальності студенти повинні навчатись весь навчальний рік.

Таким чином між типами сутностей «Група» та «Навчальний план» існує тип зв’язку «багато до багатьох»: за одним планом можуть навчатись декілька груп, та одна група протягом року навчається за двома навчальними планами. Тому потрібен асоціативний тип сутностей «План групи». Між типами сутностей «Навчальний план» та «План групи» існує тип зв’язку «один до багатьох», так само і між типами сутностей «Група» та «План групи» .

Для одного виду занять одного пункту навчального плану може бути призначено більше однієї академічної пари занять на тиждень, якщо кількість годин занять на семестр більше 2*weeks, де weeks – кількість тижнів у семестрі. Для складання розкладу занять потрібен новий асоціативний тип сутності «Заняття групи», у якому екземпляр сутності містить дані про те, для якої групи з якої дисципліни призначена академічна пара занять, якого виду заняття, та проводяться ця навчальна пара занять щотижня, або раз на два тижні.

Між типами сутностей «Вид занять дисципліни» та «Заняття групи» у загальному випадку існує тип зв’язку «один до багатьох», оскільки з одним екземпляром сутності «Вид занять дисципліни» може бути зв’язані декілька екземплярів сутності «Заняття групи», якщо для деякого виду занять деякої дисципліни призначено більше 2*weeks академічних годин на семестр.

Визначивши типи зв'язків між виявленими типами сутностей, отримаємо доповнену концептуальну модель бази даних для підсхеми «Розклад занять» (рис. 3.2).

На кожній кафедрі працює декілька викладачів, та кожен викладач, як правило, проводить декілька занять. Між типами сутностей «Кафедра» та «Викладач» існує тип зв’язку «один до багатьох».

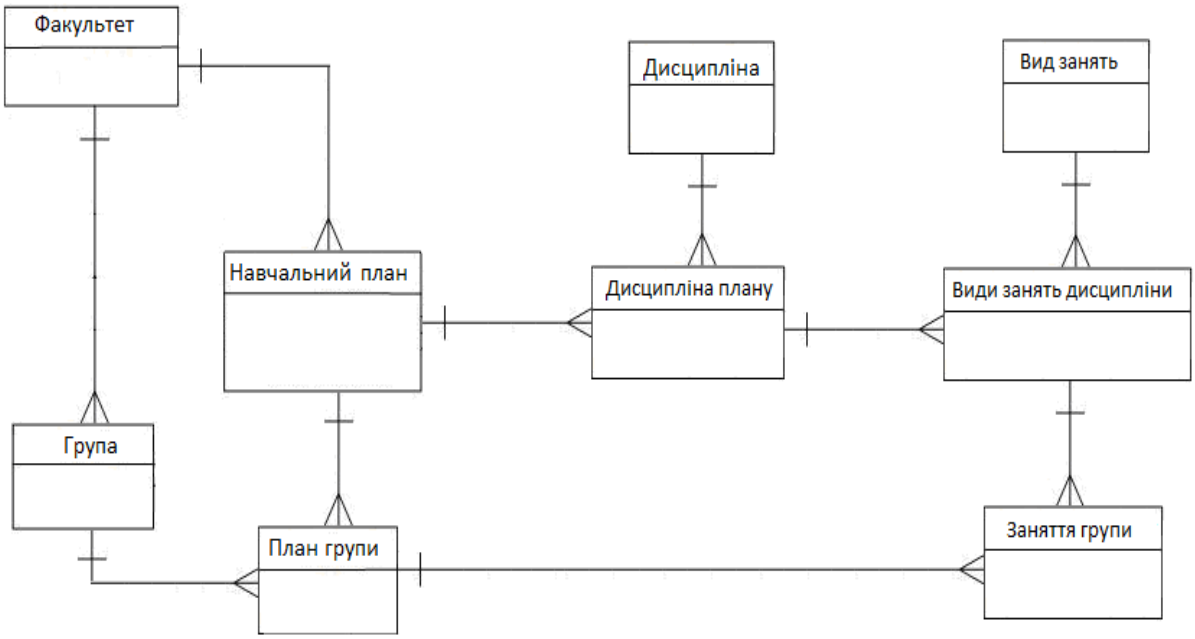

Рисунок 3.2 – Доповнена концептуальна модель БД
підсхеми «Розклад занять»

Перед розподілом занять між викладачами кафедри спочатку визначається, для яких дисциплін лекції будуть проводиться як потокові, а також склад цих академічних потоків.

У потік входить декілька груп, але деякі групи можуть входити до декількох потоків – для слухання лекцій з різних навчальних дисциплін. Таким чином, між типами сутностей «Група» і «Потік» існує тип зв’язку «багато до багатьох»; тому бажано до схеми додати ще один асоціативний тип сутності «Групи потоку».

Також на кафедрах визначається, по яких дисциплінах лабораторні заняття будуть проводитись з розподілом академічних груп на підгрупи. Потім для кожного отриманого заняття призначається викладач.

Оскільки для одного лабораторного заняття групи може бути отримано два заняття підгруп, потрібен новий похідний тип сутності «Заняття». Між типом сутностей «Заняття групи» та «Заняття» існує тип зв’язку «один до багатьох», так саме, як і між типами сутностей «Викладач» та «Заняття».

Між типами сутностей «Потік» та «Заняття» також існує тип зв’язку «один до багатьох». Визначивши типи зв'язків між виявленими типами

сутностей, отримаємо уточнену концептуальну модель бази даних для підсхеми «Розклад занять» (рис. 3.3).

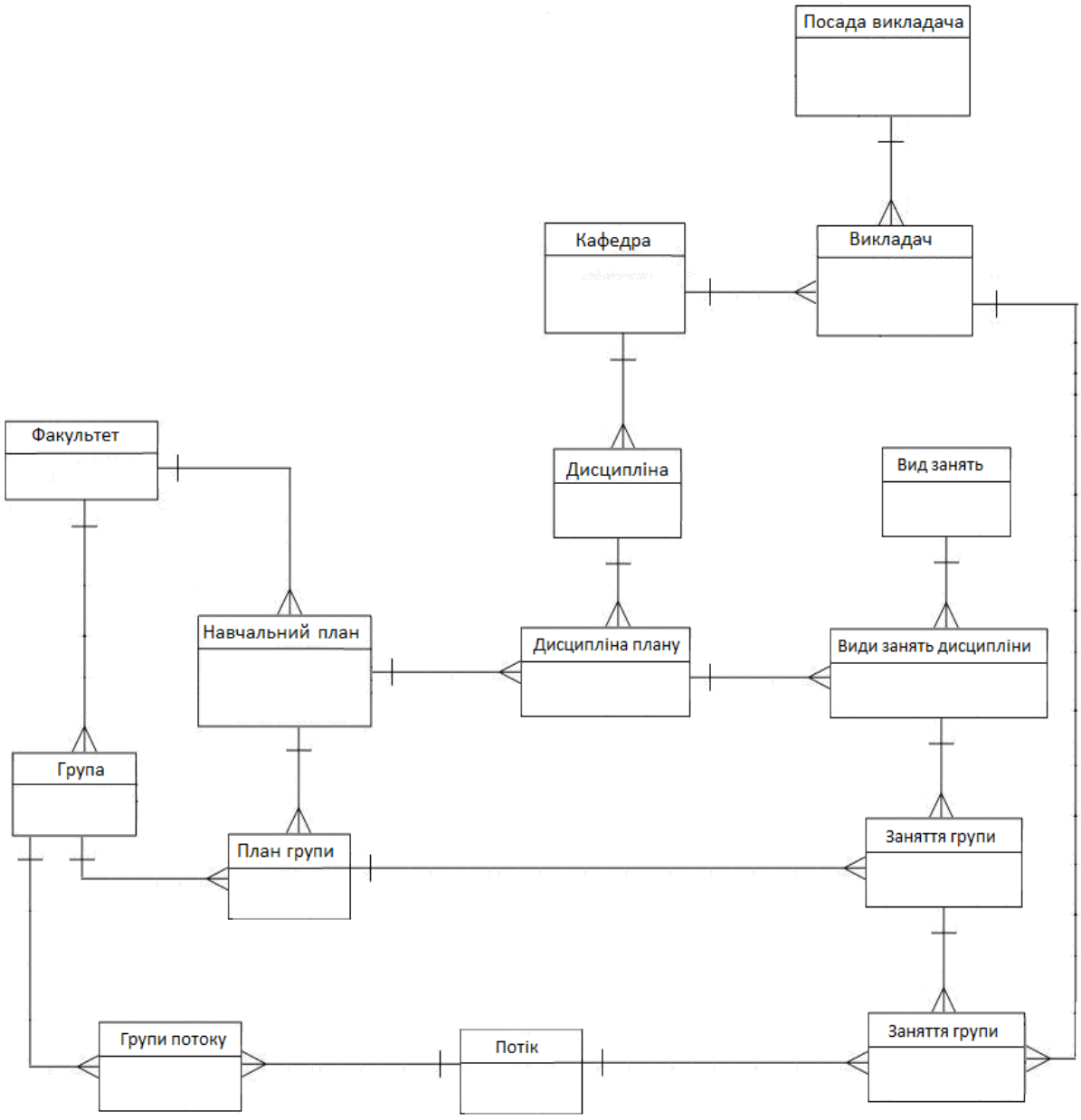

Рисунок 3.3 – Уточнена концептуальна модель БД
підсхеми «Розклад занять»

При постановці заняття у розклад можна записувати аудиторію та навчальну пару як атрибути типу сутності «Заняття». Але це не дуже зручно, тому що до постановки заняття у розклад ці атрибути повинні мати порожнє значення, також потрібно буде додати атрибут типу прапору: чи поставлене заняття у розклад. Краще додати новий похідний тип сутності «Заняття розкладу», який буде мати тільки посилання: яке заняття на яку пару

поставлене та у яку аудиторію. Для занять через тиждень також потрібно мати відомості, на якому тижні проводиться заняття.

Між типами сутностей «Заняття» та «Заняття розкладу» існує тип зв’язку «один до одного»: кожне заняття повинно бути поставлено у розклад і тільки один раз.

Для отримання зручного розкладу занять бажано для заняття призначати множину допустимих аудиторій. Тому потрібен ще похідний тип сутностей «Аудиторія заняття». Оскільки для заняття може призначатись декілька аудиторій, то між типами сутностей «Заняття» та «Аудиторія заняття» існує тип зв’язку «один до багатьох».

Кожна аудиторія може використовуватись для проведення декількох занять – на різних навчальних парах тижня. Тому між типами сутностей «Аудиторія» та «Аудиторія заняття» також існує тип зв’язку «один до багатьох».

Викладачі можуть висловлювати побажання щодо навчальних пар, на яких їм зручно проводити заняття. Ці побажання можна описати за допомогою похідного типу сутностей «Побажання викладача».

Між типами сутностей «Викладач» та «Побажання викладача» існує тип зв’язку «один до багатьох»; також і між типами сутностей «Навчальна пара» та «Побажання викладача». Після додавання перелічених типів сутностей до схеми отримуємо кінцеву концептуальну модель БД для підсхеми «Розклад занять» (рис. 3.4).

Крім складання розкладу занять до функціональних обов’язків працівників навчального відділу відноситься також перевірка планування викладачами навчального навантаження на наступний навчальний рік та перевірка звітності кафедр про виконання викладачами кафедри навчального навантаження у минулому навчальному році.

Аудиторне навчальне навантаження викладача складається з аудиторних занять, що включені до розкладу занять. Оскільки на початку нового семестру, а особливо на початку нового року можливе коригування контингенту студентів і зміна у кількості занять деяких викладачів, потрібно до схеми БД додати тип сутності «План заняття», у якому будуть зберігатись заплановані на наступний навчальний рік заняття викладачів. Зміни у навантаженні викладача будуть відображатись у типах сутностей «Заняття» та «Заняття розкладу».

Крім аудиторних занять до навчального навантаження викладача входить також керівництво курсовими роботами та курсовими проектами

студентів, а також керівництво студентами при виконанні ними кваліфікаційної випускної роботи рівня бакалавр або магістр.

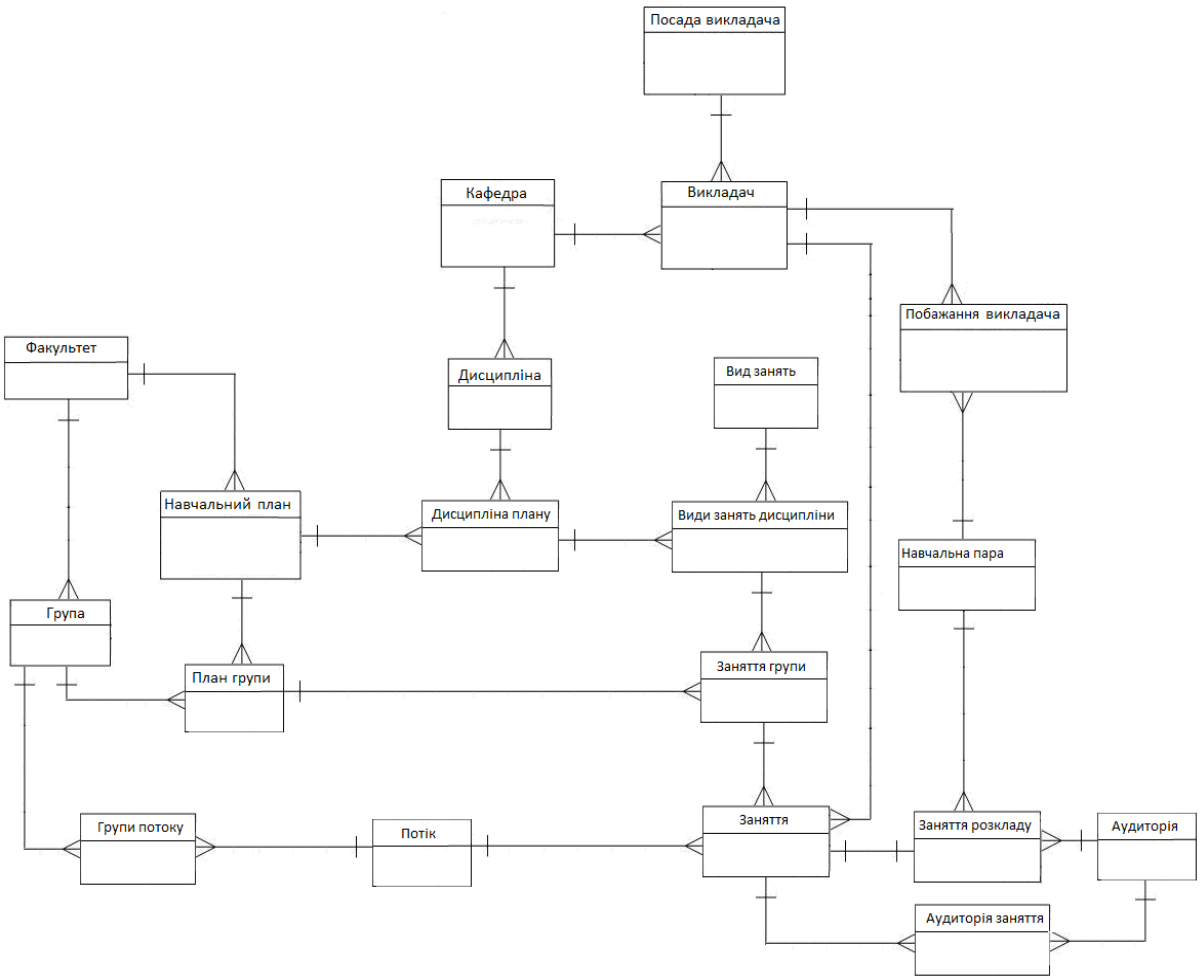

Рисунок 3.4 – Кінцева концептуальна модель БД
підсхеми «Розклад занять»

При плануванні цього навантаження викладач лише вказує кількість студентів, якими він планує керувати. Але при звітуванні про виконану роботу викладач повинен перерахувати студентів, якими він керував. Але після виконання випускної роботи студенти відраховуються з навчального закладу, та видаляються з БД. Потрібно додати у БД тип сутності, який буде зберігати архівні дані про студентів, що були відраховані з закладу з тої чи іншої причини.

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

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

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

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

У типу сутності «Навчальний план» будуть атрибути: назва, код навчального плану (ідентифікатор), код факультету, навчальний рік, семестр, кількість тижнів навчання.

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

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

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

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

У типу сутності «Дисципліна плану» будуть атрибути: код навчального плану, код дисципліни, кількість кредитів, вид семестрового контролю (іспит, залік), наявність та вид курсового проектування (проект, робота), кількість годин навчальної практики.

У типу сутності «Вид занять» будуть атрибути: назва, код виду занять (ідентифікатор).

У типу сутності «Вид занять дисципліни» будуть атрибути: код навчального плану, код дисципліни, код виду занять, кількість годин занять (за семестр).

У типу сутності «Заняття групи» будуть атрибути: код заняття (ідентифікатор), код дисципліни, код навчального плану, код виду занять, код групи, номер заняття на протязі тижня (одного виду однієї дисципліни), чи проводиться щотижня.

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

У типу сутності «Заняття розкладу» будуть атрибути: код заняття (ідентифікатор), ідентифікатор аудиторії, код навчальної пари, позначка тижня проведення (парний, непарний, кожен тиждень).

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

3.2.3 Визначення атрибутів, що є потенційними і первинними ключами

У всіх базових типів сутностей первинним ключем буде атрибут з назвою «ідентифікатор». У типів сутностей, що за змістом є таблицями-довідниками: «Факультет», «Кафедра», «Дисципліна», «Посада викладача», «Вид занять» – потенційним ключем (унікальним) є атрибут «назва».

Похідні та асоціативні типи сутностей повинні мати складові потенційні (унікальні) ключі, до складу яких входять атрибути, що є зовнішніми ключами. Для спрощення розрахунків у якості первинних ключів в цих таблицях можна призначати додаткові атрибути цілого типу.

3.2.4 Визначення відповідності концептуальної моделі транзакціям користувачів

У розділі 3.2.1 перелічені транзакції основних груп користувачів ІС «Університет», що стосуються процесів розрахунку навчального навантаження викладачів та процесу складання розкладу занять.

Розглянемо транзакції з точки зору можливості виконання їх .

1) Отримати від факультетів робочі навчальні плани на наступний навчальний рік. Для цієї транзакції потрібно зробити вибірку даних з зв’язаних таблиць: «Факультет» – «Навчальний план» – «Дисципліна плану» – «Вид заняття дисципліни» – «Вид занять».
2) Отримати від факультетів контингенти студентів на наступний семестр. Для цієї транзакції потрібно зробити вибірку даних з зв’язаних таблиць «Факультет» – «Група».

3) Отримати від факультетів прикріплення академічних груп до спеціальностей та освітньо-професійних програм (ООП) навчання. Для цієї транзакції потрібно зробити вибірку даних з зв’язаних таблиць «Факультет» – «Навчальний план» – «План групи» – «Група».
4) Отримати від методичного відділу списки дисциплін, закріплених за кафедрами на наступний навчальний рік. Для цієї транзакції потрібно зробити вибірку даних з зв’язаних таблиць «Кафедра» – «Дисципліна».
5) Отримати від кафедр список потокових лекцій по дисциплінах кафедри. Для цієї транзакції не передбачений похідний тип сутностей, що зберігає наступні відомості: лекції по яких дисциплінах яких навчальних планів читаються на потоці, та якому саме потоку. Потрібно додати тип сутності «Лекції потоку» з вказаним набором атрибутів. Тоді для цієї транзакції потрібно зробити вибірку даних з зв’язаних таблиць «Кафедра» – «Дисципліна» – «Дисципліна плану» – «Вид заняття дисципліни» – «Заняття групи» – «Лекції потоку» – «Заняття».
6) Отримати від кафедр список лабораторних занять по дисциплінах кафедр, які проводяться з поділом груп на підгрупи. Для цієї транзакції також не передбачений похідний тип сутностей, що зберігає наступні відомості: лабораторні заняття по яких дисциплінах яких навчальних планів проводяться з поділом груп на підгрупи. Потрібно додати тип сутності «Заняття підгруп» з вказаним набором атрибутів. Тоді для цієї транзакції також потрібно зробити вибірку даних з зв’язаних таблиць «Кафедра» – «Дисципліна» – «Дисципліна Плану» – «Вид заняття дисципліни» – «Заняття групи» – «Заняття підгруп» – «Заняття».
7) Отримати від кафедр розподіл між викладачами кафедри аудиторних занять по дисциплінах кафедри на наступний семестр. Для цієї транзакції також не передбачений похідний тип сутностей, що зберігає відомості: яке заняття проводить який викладач. Потрібно додати тип сутності «Заняття викладача» з вказаним набором атрибутів. Тоді для цієї транзакції також потрібно зробити вибірку даних з зв’язаних таблиць «Кафедра» – «Викладач» – «Заняття викладача» – «Заняття».

8) Відновити дані по аудиторному фонду університету на наступний семестр. Для виконання цієї транзакції для користувачів групи «Працівник навчального відділу» повинні бути призначені права додавання, оновлення та усунення записів в таблиці «Аудиторія».
9) Отримати від кафедр перелік побажань та вимог викладачів до розкладу занять на наступний семестр. Для цієї транзакції потрібно зробити вибірку даних з зв'язаних таблиць «Кафедра» – «Викладач» – «Побажання викладача» – «Навчальна пара».
10) Скласти чорновий розклад занять на наступний семестр. Для виконання цієї транзакції потрібно використання програми виду «Автоматичне складання розкладу занять» або «Автоматизоване складання розкладу занять».
11) Передати чорновий розклад занять на кафедри для можливого корегування розкладу. Для виконання цієї транзакції потрібно мати у базі даних інформаційної системи таблицю допусків різних груп користувачів до різної інформації у БД. Якщо така таблиця є, то потрібно для таблиці «Заняття розкладу» встановити для атрибут «Готовність розкладу» значення «Чорновий розклад». Групам користувачів «Працівник кафедри» та «Викладач» встановити право переглядати розклад занять, коли встановлене значення «Чорновий розклад».
12) Отримати від кафедр дані для корегування розкладу занять. Для виконання цієї транзакції у таблиці допусків груп користувачів до інформації у БД потрібно мати атрибути, які містять дані, що інформація для складання розкладу занять кафедрою оновлена, та дату оновлення даних для розкладу занять. Якщо кафедра не змінює дані для розкладу занять, працівник кафедри повинен просто змінити дату оновлення даних для своєї кафедри.
13) Скласти остаточний розклад занять на наступний семестр. Для виконання цієї транзакції також використовується програма виду «Автоматичне складання розкладу занять» або «Автоматизоване складання розкладу занять» з оновленими даними.
14) Отримати від кафедр повне заплановане навчальне навантаження викладачів на наступний навчальний рік та затвердити його. Повне навчальне навантаження включає в себе не тільки аудиторне навантаження, а також виконання не аудиторних видів навчальної роботи: керування курсовими та дипломними проектами,

проведення консультації з дисципліни провідним викладачем, проведення іспитів та консультацій перед іспитами, перевірку результатів проведення контролюючих заходів. Для підрахунку цих видів навчального навантаження викладачів потрібно у базу даних додати таблицю-довідник з переліком видів не аудиторного навчального навантаження та перерахуванням їх у академічні часи загального навчального навантаження викладача.

15) Перевірити виконання викладачами навчального навантаження у попередньому навчальному році. При наявності у БД типу сутності «План заняття», що містить запланований розподіл занять між викладачами університету, порівняння запланованого та виконаного навантаження відбувається вибіркою з цієї таблиці, та таблиці «Заняття», зв’язаної з таблицею «Заняття розкладу».

Після додання у БД таблиці допусків різних груп користувачів до інформації у БД, таблиці-довідника з переліком видів не аудиторного навчального навантаження, а також необхідних похідних типів сутностей «Лекції потоку» та «Заняття підгруп», всі перелічені транзакції можна буде виконати (рис. 3.5).

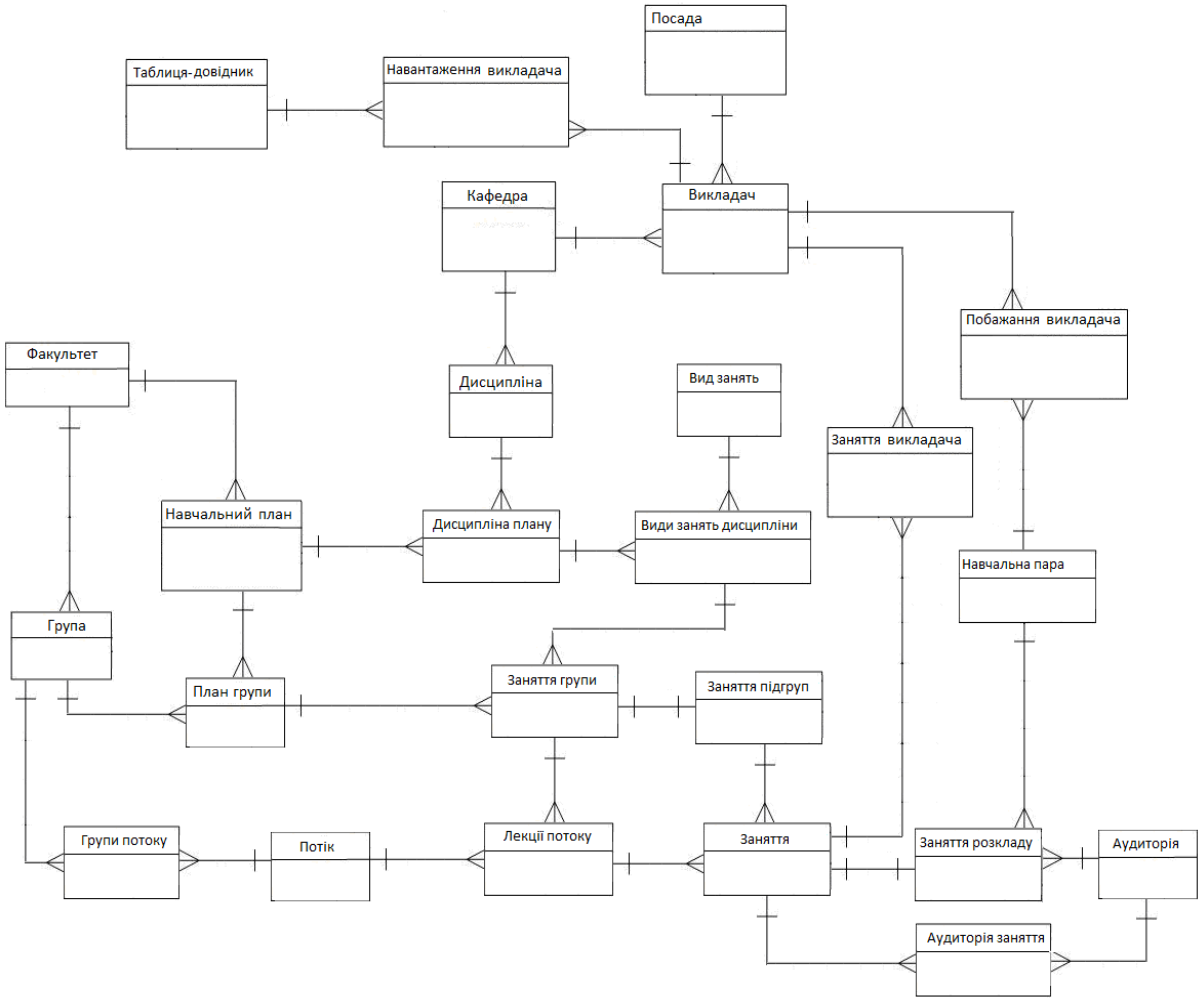

Рисунок 3.5 – Остаточна концептуальна модель БД підсхеми «Розклад занять» для підсистеми «Навчальний відділ»

Для транзакцій, призначених групі користувачів «Працівник навчального відділу» визначені сутності, їх атрибути і зв’язки між ними. Отже, отримана концептуальна модель коректна у контексті підсистеми «Навчальний відділ».

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

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

4 ВИБІР СУБД ТА СЕРЕДОВИЩА РОЗРОБКИ ЗАСТОСУВАННЯ

4.1 Вибір СУБД

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

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

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

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

4.2 Вибір середовища розробки застосування та мови програмування

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

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

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

Microsoft Visual Studio – лінійка продуктів компанії Майкрософт, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів. Дані продукти дозволяють розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-

служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Microsoft Silverlight.

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

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

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

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

Visual Studio 2015 оснащена всіма удосконаленими засобами тестування для забезпечення стабільної якості коду. Розробники застосувань можуть використовувати кодовані тести інтерфейсу користувача, за допомогою яких можна автоматизувати тестування інтерфейсу веб-додатків і додатків Windows, засоби ручного тестування, функцію Test Professional і інші корисні функції.

Розробка баз даних вимагає тієї ж ретельності і уваги що і розробка додатків. У Visual Studio 2015 це враховується: користувачам надаються надійні засоби розробки та управління змінами, які дозволяють синхронізувати додаток і базу даних. Таким чином програмне середовище розробки застосувань компанії Microsoft Visual Studio 2015 є дуже вдалим вибором у якості середовища розробки застосувань для баз даних.

5 ЛОГІЧНЕ І ФІЗИЧНЕ ПРОЕКТУВАННЯ БД

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

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

5.1 Логічне проектування

Розроблена в третьому розділі модель вже майже є логічною моделлю реляційної бази даних. Як видно з концептуальної моделі, представленої на рис.3.5, ніде немає типів зв’язків «багато до багатьох», які не допустимі у реляційної бази даних.

Таким чином, набір сутностей, визначених при концептуальному проектуванні, перетворюється в набір відношень при логічному проектуванні по реляційній моделі.

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

На рис. 5.1 приведена ER-діаграма БД інформаційної системи для підсхеми «Навчальний відділ».

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

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

Як видно з рис. 3.5 та рис. 5.1, діаграма бази даних суттєво відрізняється від діаграми у вигляді ієрархічного дерева. Зв’язки між відношення у базі даних у багатьох випадках створюють коло.

У цих випадках потрібно намагатись уникати встановлення для зовнішніх ключів можливості каскадного оновлення або усунення записів у підлеглих відношеннях.

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

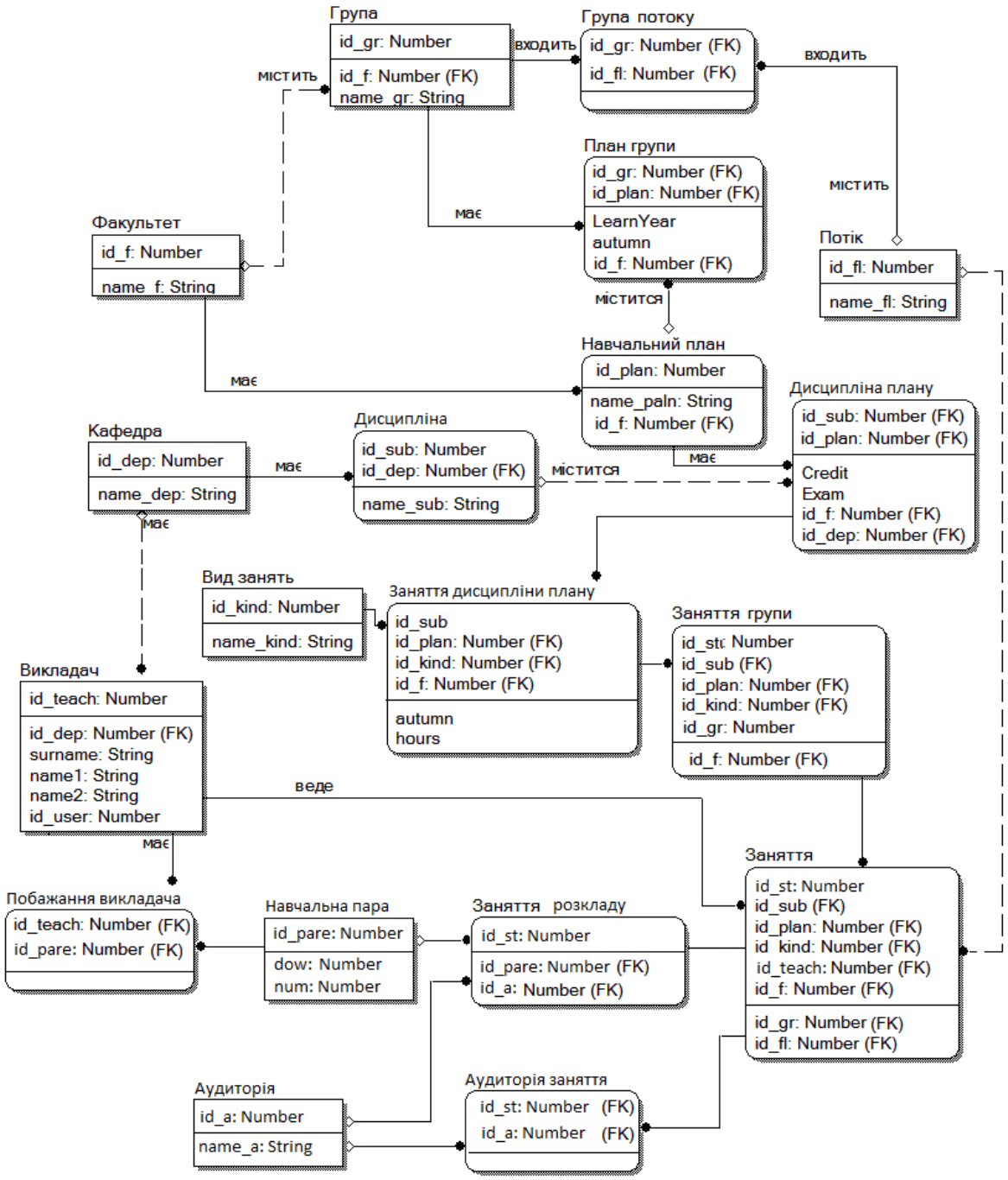

Рисунок 5.1 – ER-діаграма БД інформаційної системи для підсхеми «Навчальний відділ»

5.2 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 інформаційній системі, що розробляється, передбачається велика кількість користувачів з правами маніпулювання даними в таблицях бази даних. Такі права будуть мати майже всі групи користувачів, крім груп Студент та Гість. Наприклад, користувачів групи «Працівник кафедри» буде декілька, щонайменше одна особа від кожної кафедри. У подібних випадках небажано мати первинні ключі з властивістю автоматичної нумерації для таблиць, права на додавання і видалення даних в яких мають багато користувачів.

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

6 СЕРВЕРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

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

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

Для тонких клієнтів не бажано генерувати SQL-код команд на маніпулювання записів у таблицях БД. Ці застосування повинні лише передавати до серверу виклик збережених процедур. А вже збережені процедури будуть виконувати необхідні дії по маніпулюванню даними у таблицях БД. У клієнтському застосуванні після виклику збережених процедур, що змінюють вміст таблиць БД потрібно лише оновити компоненти форми програми, що от ображають інформацію у таблицях.

Крім збережених процедур, що виконуватимуть роль тригерів для схеми «Навчальний відділ» потрібно додати процедуру, що заповнює таблицю ПланЗаняття, яка зберігає дані по запланованому на навчальний рік аудиторному навантаженню викладачів.

Також потрібні процедури, що перевіряють готовність вихідних даних для складання розкладу занять на заданий навчальний рік.

Навчальні дані поступають від декількох груп користувачів. Спочатку деканати факультетів повинні розробити робочі навчальні плани на наступний навчальний рік.

Потім методичний відділ університету повинен затвердити розроблені навчальні плани. По затвердженим навчальним планам визначаються заняття академічних груп, які прикріплюються до деякого навчального плану.

Коли інформація про навчальні плани, контингенти студентів та прикріплення груп до навчальних планів стає доступною у БД працівникам кафедр, вони розподіляють навантаження по дисциплінах кафедри між викладачами.

Після розподілу всіх занять між викладачами можна створювати розклад занять на наступний семестр.

Таким чином, серед серверного програмного забезпечення повинні бути збережені процедури, що перевіряють повноту вихідних даних до програми складання розкладу занять.

Процедура AuditAdd додає запис у таблицю Auditory:

```
CREATE procedure AuditAdd
  @id_aud smallint output,
  @korpus tinyint,
  @name_aud varchar (6),
  @kol tinyint,
  @k tinyint output,
  @msg varchar (50) output
```

Параметри процедури:

@id_aud – ідентифікатор аудиторії;

@korpus – номер навчального корпусу, де знаходиться аудиторія;

@name_mod – назва аудиторії;

@kol – місткість аудиторії, кількість посадкових місць;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура AuditAdd перевіряє, чи вже є в БД запис про аудиторію з заданою назваю у заданому навчальному корпусі. Якщо запис про таку аудиторію є, то генерується повідомлення про помилку.

Якщо такого запису немає, то в таблицю Auditory додається запис зі значеннями атрибутів: name_aud = @name_aud, korpus = @korpus, kol = @kol. Процедура визначає для доданого запису значення первинного ключа таблиці Auditory – ідентифікатор @id_aud, – та повертає його значення.

Процедура AuditEdit змінює запис у таблиці Auditory:

```
CREATE procedure AuditEdit
  @id_aud smallint,
  @korpus tinyint,
  @name_aud varchar (6),
  @kol tinyint,
  @k tinyint output,
  @msg varchar (50) output
```

Параметри процедури:

@id_aud – ідентифікатор аудиторії;

@korpus – номер навчального корпусу, де знаходиться аудиторія;

@name_mod – назва аудиторії;

@kol – місткість аудиторії, кількість посадкових місць;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура AuditEdit перевіряє, чи вже є в таблиці Auditory запис, що задовольняє наступній умові: name_aud = @name_aud and korpus =@korpus id_aud != @id_aud.

Якщо запис про таку аудиторію є, то генерується повідомлення про помилку.

Якщо такого запису немає, то в таблиці Auditory у записа з ключем id_aud = @id_aud змінюються значення атрибутів name_aud, korpus, kol, на значення параметрів процедури @name_aud, @korpus, @kol відповідно.

Процедура AuditDel усуває запис з таблиці Auditory:

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

Параметри процедури:

@id_aud – ідентифікатор аудиторії;

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

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Якщо @m=0, то процедура AuditDel перевіряє, чи є у підлеглих таблицях StudyAuditory та IsInTT записи, що звязані по ключу id_aud з записом таблиці Auditory, що видаляється. Якщо такі записи є, то генерується повідомлення про помилку. Якщо таких записів немає, то запис з ключом id_aud = @id_aud видаляється з таблиці Auditory.

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

Процедура FillStudyPlan заповнює записами таблицю StudyPlan:

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

Параметри процедури:

@LearnYear – навчальний рік (осіннього семестру);

@m – прапор, чи видаляти наявні записи з таблиці;

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Якщо @m=0, то процедура FillStudyPlan перевіряє, чи вже є в таблиці StudyPlan записи з атрибутом: LearnYear = @LearnYear.

Якщо такі записи є, то генерується повідомлення про помилку і в параметр @msg записується кількість записів таблиці StudyPlan з вказаним значенням атрибуту LearnYear, та кількість записів таблиці Study з тим же значенням атрибуту LearnYear.

Якщо таких записів немає, то з таблиці Study у таблицю StudyPlan копіюються всі записи з атрибутом: LearnYear = @LearnYear.

Якщо @m=1, то процедура FillStudyPlan видаляє з таблиці StudyPlan всі записи з атрибутом LearnYear = @LearnYear. Потім з таблиці Study у таблицю StudyPlan копіюються всі записи з атрибутом: LearnYear = @LearnYear. StudyPlan не редагується, коли протягом навчального року змінюються контингенти студентів, кількість груп у потоках лекцій, поділ груп на підгрупи при проведенні лабораторних занять. Всі ці зміни відображаються у таблицях Study та IsInTT.

Таблиця StudyPlan відображає заплановане аудиторне навантаження викладачів, а таблиця Study – виконане навчальне аудиторне навантаження викладачів.

Процедура CheckFacultet перевіряє, які факультети надали неповні дані для складання розкладу занять на заданий навчальний рік:

```
CREATE PROC CheckFacultet
@LearnYear smallint,
  @k tinyint output,
  @msg varchar (50) output
```

Параметри процедури:

@LearnYear – навчальний рік (осіннього семестру);

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура CheckFacultet перевіряє, чи є в таблиці LearnPlan серед кортежів з атрибутов LearnYear = @LearnYear ті, що має значення атрибуту approve != 1 (не затверджені методичним відділом навчальні плани заданого навчального року). Якщо такі записи є, то параметру @k присвоюється значення 1, а в повідомлення про помилку виводиться перелік факультетів з незатвердженими навчальними планами.

Якщо всі навчальні плани затверджені, процедура CheckFacultet перевіряє, чи всі академічні групи з запланованим ненульовим контингентом

прикріплені до навчальних планів. Якщо це не так, то параметру @k присвоюється значення 2, а в повідомлення про помилку виводиться перелік факультетів та їх академічних груп, не прикріплених до навчальних планів.

Процедура CheckDepartment перевіряє, які кафедри надали неповні дані для складання розкладу занять на заданий навчальний рік:

```
CREATE PROC CheckDepartment
@LearnYear smallint,
 @k tinyint output,
 @msg varchar (50) output
```

Параметри процедури:

@LearnYear – навчальний рік (осіннього семестру);

@k – код помилки, вихідний параметр;

@msg – вихідний параметр, повідомлення про помилку.

Процедура CheckDepartment перевіряє, чи є в таблиці Study серед кортежів з атрибутов LearnYear = @LearnYear ті, що мають порожнє значення атрибуту id_teach (не призначені викладачі для проведення заняття). Якщо такі записи є, то параметру @k присвоюється значення 1, а в повідомлення про помилку виводиться перелік кафедр з незатвердженими навчальними планами.

Якщо всі заняття розподілені між викладачами, збережена процедура CheckDepartment перевіряє, чи для всіх занять заданий список допустимих аудиторій. Якщо це не так, то параметру @k присвоюється значення 2, а в повідомлення про помилку виводиться перелік кафедр та їх занять, для яких не призначені аудиторії.

7 ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ

7.1 Загальні відомості

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

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

7.2 Функціональне призначення

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

Програма призначена для перегляду в зручному вигляді даних, необхідних для складання розкладу занять: навчальних планів, контингенту студентів, списків закріплення академічних груп за спеціальностями, аудиторного фонду ЗВО, навантаження викладачів.

Також дозволяє оновлювати ті дані, на маніпулювання якими користувач має право.

7.3 Керівництво програміста

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

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

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

public Form1()

Конструктор класу Form1 ініціалізує компоненти форми, заповнює компоненти форми програми, що є випадними списками: depComboBox, depComboBox2, depComboBox3, depComboBox4, – списками кафедр університету, які отримує як рядки запиту до БД. Також заповнює випадні списки: facComboBox, facComboBox2, facComboBox3, – списками факультетів університету, які отримує як рядки запиту до БД.

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

Методи класу форми Form1.

Метод init_commands_delAudCmd:

private void init_commands_delAudCmd()

Призначений для ініціалізації команди виклику збереженої процедури AuditDel:

Метод init_commands_addAudCmd:

private void init_commands_addAudCmd ()

Призначений для ініціалізації команди виклику збереженої процедури AuditAdd.

Метод init_commands_editAudCmd: ()

Призначений для ініціалізації команди виклику збереженої процедури AuditEdit.

Метод init_commands_ChkFacCmd:

private void init_commands_ChkFacCmd ()

Призначений для ініціалізації команди виклику збереженої процедури CheckFacultet.

Метод init_commands_ChkDepCmd:

private void init_commands_ChkDepCmd ()

Призначений для ініціалізації команди виклику збереженої процедури CheckDepartmentt.

Метод init_commands_fillStdPlCmd:

private void init_commands_fillStdPlCmd ()

Призначений для ініціалізації команди виклику збереженої процедури FillStudyPlan.

Метод init_commands_chkFacCmd:

private void init_commands_chkFacCmd ()

Призначений для ініціалізації команди виклику збереженої процедури CheckFacultet.

Метод init_commands_chkDepCmd:

private void init_commands_chkDepCmd ()

Призначений для ініціалізації команди виклику збереженої процедури CheckDepartment.

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

private void fillDepComboBox()

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

Метод fillFacComboBox заповнює випадні списки факультетів університету значеннями запиту до БД:

private void fillFacComboBox ()

Метод fillFacComboBox отримує з запиту до БД список факультетів університету, та заповнює цим списком три компоненти типу ComboBox на вкладках форми: facComboBox, facComboBox2, facComboBox3.

Метод fillTeachComboBox заповнює випадний список викладачів заданої кафедри ЗВО:

private void fillTeachComboBox(short id_dep)

Параметр методу:

id_dep – ідентифікатор кафедри, де працюють викладачі.

Метод fillTeachComboBox отримує з запиту до БД список викладачів університету: прізвище та ініціали, – та заповнює цим списком компонент типу ComboBox teachComboBox1 на третій вкладці форми.

Метод fillTeachComboBox2 заповнює випадний список викладачів заданої кафедри ЗВО у іншому форматі:

private void fillTeachComboBox2(short id_dep)

Параметр методу:

id_dep – ідентифікатор кафедри, де працюють викладачі.

Метод fillTeachComboBox2 отримує з запиту до БД список викладачів університету: посада, прізвище та ініціали, – та заповнює цим списком компонент teachComboBox на шостій вкладці форми.

Метод fillGroupComboBox заповнює значеннями з запиту до БД випадний список з переліком академічних груп факультету:

private void fillGroupComboBox (short id_f)

Параметр методу:

id_f – ідентифікатор викладача.

Метод fillGroupComboBox отримує з запиту до БД список груп заданого факультету університету: назва, кількість студентів, – та заповнює цим списком компонент groupComboBox.

Метод fillAudComboBox заповнює значеннями з запиту до БД випадний список з переліком навчальних аудиторій корпусу університету:

private void fillAudComboBox (byte korpus)

Параметр методу:

korpus – номер навчального корпусу.

Метод fillAudComboBox отримує з запиту до БД список аудиторій заданого навчального корпусу університету: назва, кількість місць, – та заповнює цим списком компонент audComboBox.

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

Обробник події натискання на кнопку yearButton:

private void yearButton_Click(object sender, EventArgs e)

Метод yearButton_Click є обробником події натискання на кнопку «ОК» біля текстового поля для вводу номеру навчального року. При натисканні на цю кнопку з рядку тексту у випадному списку вибирається перше число. Це числове значення записується у змінну класу форми Form1 LearnYear. Змінна LearnYear використовується як параметр у запитах до БД.

Обробник події вибору пункту випадного списку depComboBox:

private void depComboBox_SelectedIndexChanged(object sender,
EventArgs e)

private void depComboBox2_SelectedIndexChanged(object sender,
EventArgs e)

private void depComboBox3_SelectedIndexChanged(object sender,
EventArgs e)

private void depComboBox4_SelectedIndexChanged(object sender,
EventArgs e)

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

Обробник події вибору пункту випадного списку facComboBox:

private void facComboBox_SelectedIndexChanged(object sender,
EventArgs e)

private void facComboBox2_SelectedIndexChanged(object sender,
EventArgs e)

private void facComboBox3_SelectedIndexChanged(object sender,
EventArgs e)

Перераховані методи є обробниками події вибору пункту випадного списку «Факультет». Цей метод зчитує ідентифікатор вибраної кафедри та записує у змінну id_f. Змінна id_f використовується як параметр запитів до БД, які виводять дані по факультету на компонент типу DataGridView на формі програми. Також змінна id_f є параметром запиту, за допомогою якого випадний список groupComboBox заповнюється назвами груп вибраного факультету.

Обробник події вибору пункту випадного списку teachComboBox:

private void teachComboBox_SelectedIndexChanged(object sender,
    EventArgs e)

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

Обробник події вибору пункту випадного списку groupComboBox:

private void groupComboBox_SelectedIndexChanged(object sender,
    EventArgs e)

Метод є обробником події вибору пункту випадного списку академічних груп факультету. Цей метод зчитує в змінну id_gr ідентифікатор вибраної у списку групи.

Обробник події натискання на кнопку addAudButton:

private void addAudButton_Click(object sender, EventArgs e)

Метод addThemeButton_Click є обробником події натискання на кнопку «Добавить аудиторию в список» на вкладці форми «Аудитории». Цей метод робить недоступними кнопки на сторінці та робить видимими приховані елементи форми, які використовуються для введення даних при додаванні запису у таблицю аудиторій навчального корпусу.

Обробник події натискання на кнопку editAudButton:

private void editAudButton_Click(object sender, EventArgs e)

Метод editAudButton_Click є обробником події натискання на кнопку «Изменить данные по аудитории» на вкладці «Аудитории». Цей метод також робить недоступними кнопки на сторінці та робить видимими приховані елементи форми, які використовуються для оновлення запису.

Обробник події натискання на кнопку delAudButton:

private void delAudButton_Click(object sender, EventArgs e)

Цей метод є обробником події натискання на кнопку «Удалить аудиторию из списка» також на вкладці «Аудитории». Цей метод виводить на форму запитання, чи дійсно користувач хоче видалити з бази даних запис

про вибрану аудиторії вибраного навчального корпусу. Якщо користувач вибирає відповідь «OK», то викликається збережена процедура бази даних AuditDel для видалення запису в таблиці Auditory.

Обробники події натискання на перемикачі на вкладках форми:

private void checkBox1_CheckedChanged(object sender, EventArgs e)

private void checkBox2_CheckedChanged(object sender, EventArgs e)

private void checkBox3_CheckedChanged(object sender, EventArgs e)

private void checkBox4_CheckedChanged(object sender, EventArgs e)

private void checkBox5_CheckedChanged(object sender, EventArgs e)

Ці методи змінюють значення змінної autumn (осінній чи весняний семестр у запитах до БД).

Обробники події зміни поточного рядка у компоненті dataGridView2

private void dataGridView2_CellEnter(object sender,

DataGridViewCellEventArgs e)

Цей метод зчитує в змінну id_plan значення ідентифікатору навчального плану у поточному рядку табличного компоненту на вкладці формі dataGridView2. Змінна id_plan є параметром запиту, який вибирає дані для заповнення другого табличного компоненту dataGridView3.

7.4 Посібник користувача

При запуску програми Teacher.exe на екрані з’являється форма програми (рис. 7.1).

Спочатку користувачеві потрібно вибрати навчальний рік, з даними по якому він збирається працювати.

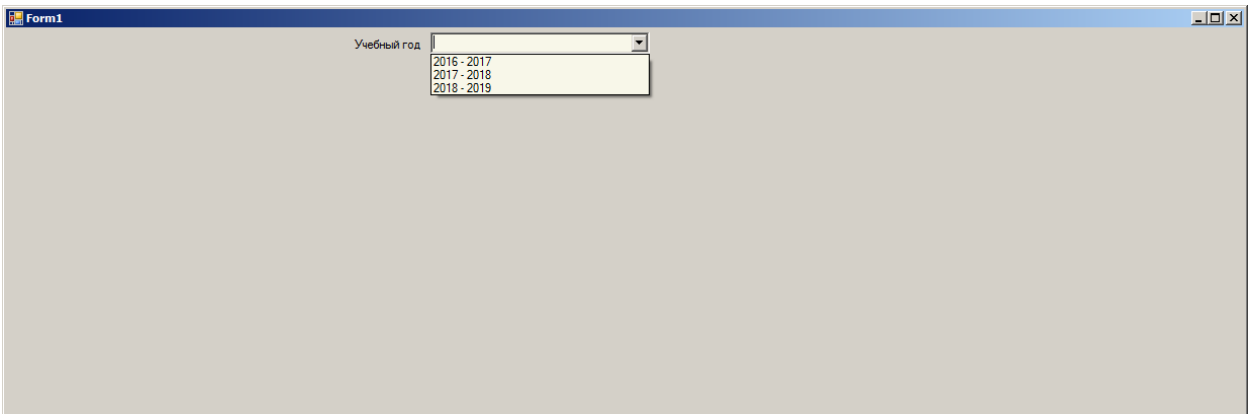

Рисунок 7.1 – Початок роботи з програмою

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

На першій вкладці форми – «Учебные планы» – користувач повинен вибрати факультет, навчальні плани якого він хоче переглянути, із списку факультетів університету (рис. 7.2).

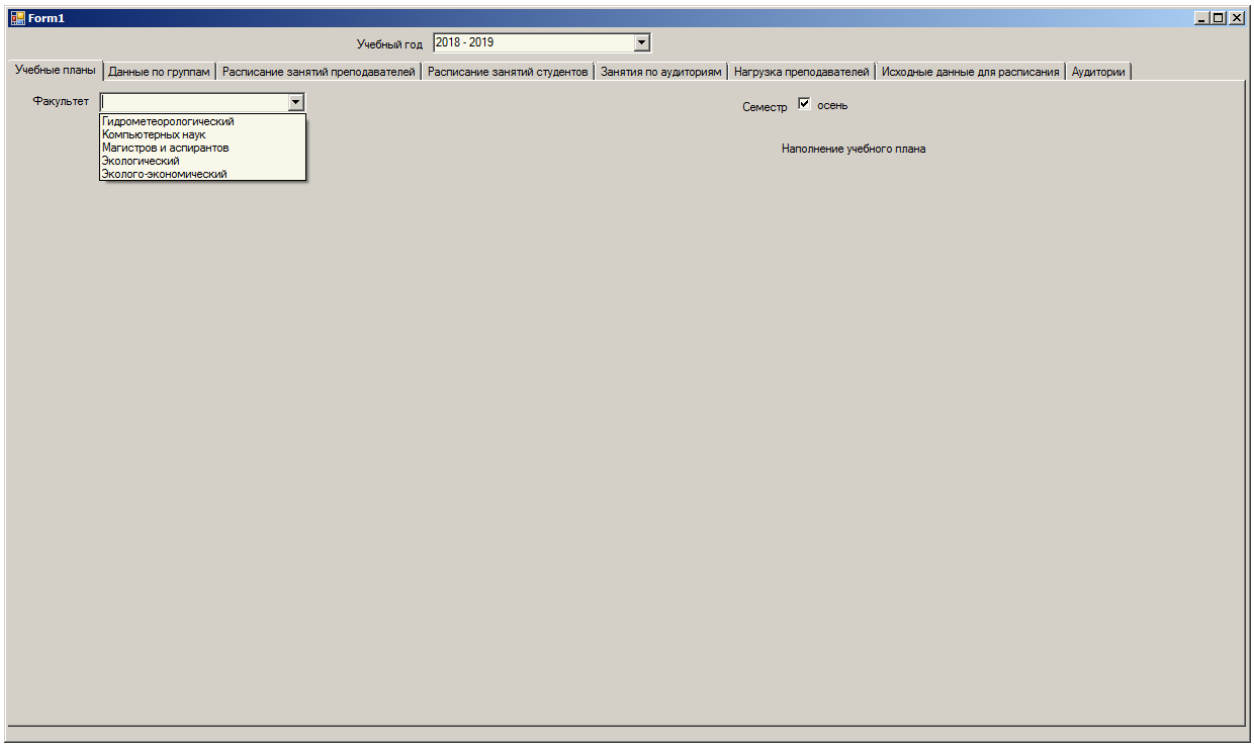

Рисунок 7.2 – Вибір користувачем факультету із списку

Після вибору факультету на формі стають видимими два табличних компонента. У лівому компоненту виводиться список навчальних планів вибраного факультету за вибраний навчальний рік (рис. 7.3).

Плани виводяться для осіннього семестру. Щоб отримати навчальні плани факультету на весняний семестр вибраного навчального року, потрібно клацнути по перемикачеві «Семестр» зверху вкладки.

Як видно з рис. 7.3 у лівому табличному компоненті виводяться тільки назви навчальних планів, номер курсу, номер навчального семестру та кількість навчальних тижнів у цьому семестрі.

Вміст навчального плану, тобто перелік дисциплін з їх наповненням, виводиться у правому табличному компоненті цієї вкладки. Вміст виводиться для поточного рядка у лівому компоненті, тобто виділеного рядка таблиці ліворуч (рис. 7.4).

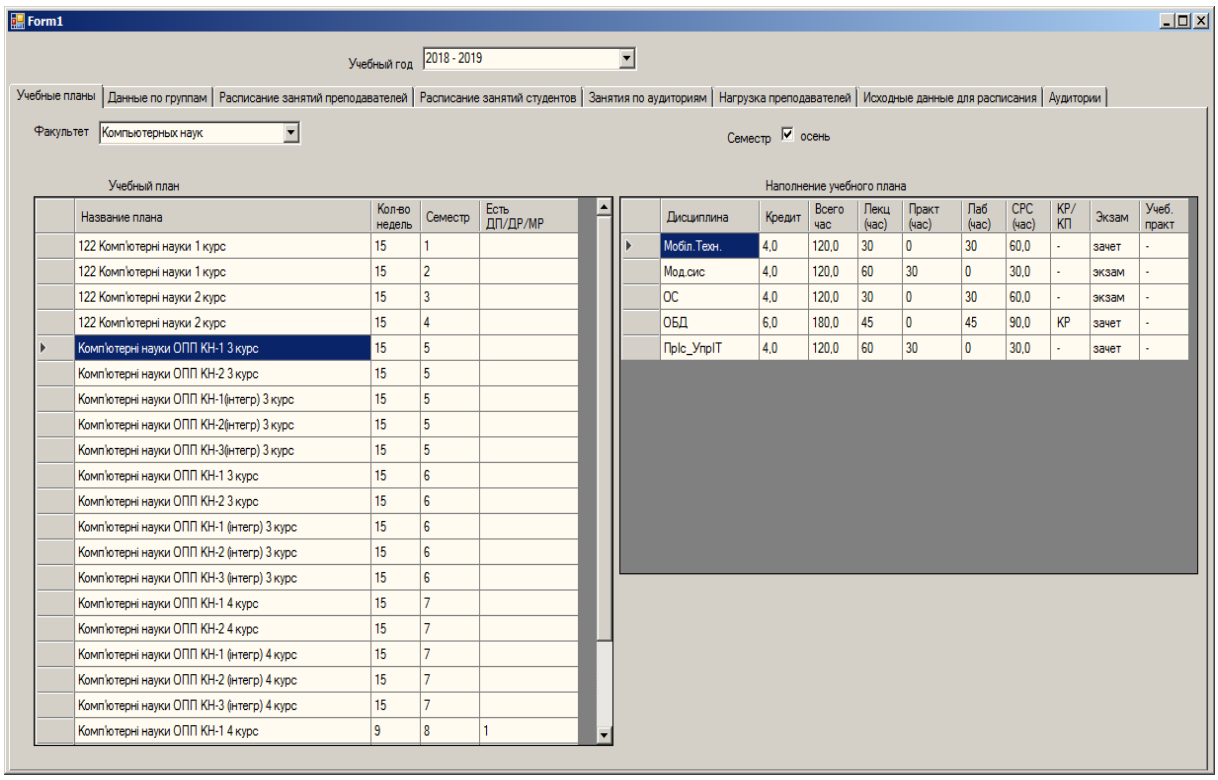

Рисунок 7.3 – Виведення переліку навчальних планів вибраного факультету вибраного навчального року

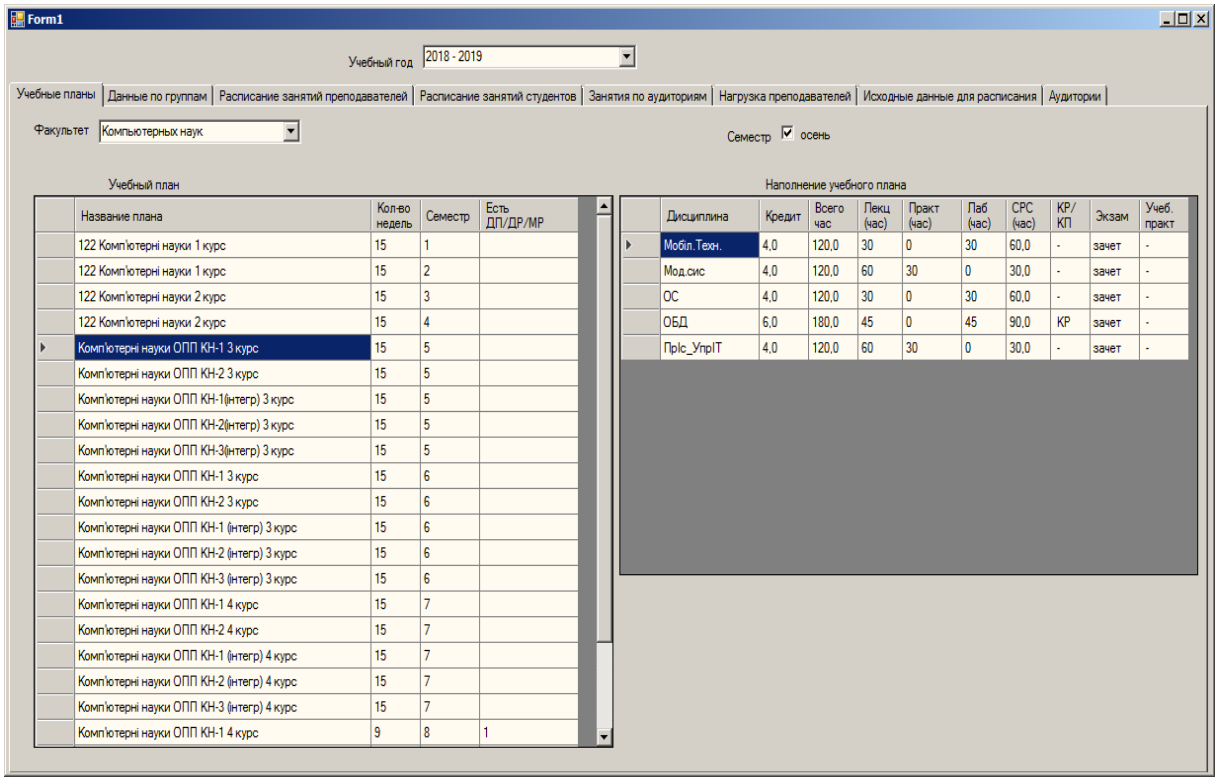

Рисунок 7.4 – Зміна вмісту правої таблиці форми при зміні поточного рядка у таблиці ліворуч

Друга вкладка форми має назву «Данные по группам». Для отримання цих даних також потрібно вибрати факультет (рис. 7.5). Після цього на формі з’являються два табличні компоненти, у які виводять відомості про прикріплення груп до навчальних планів та контингент студентів (рис. 7.6).

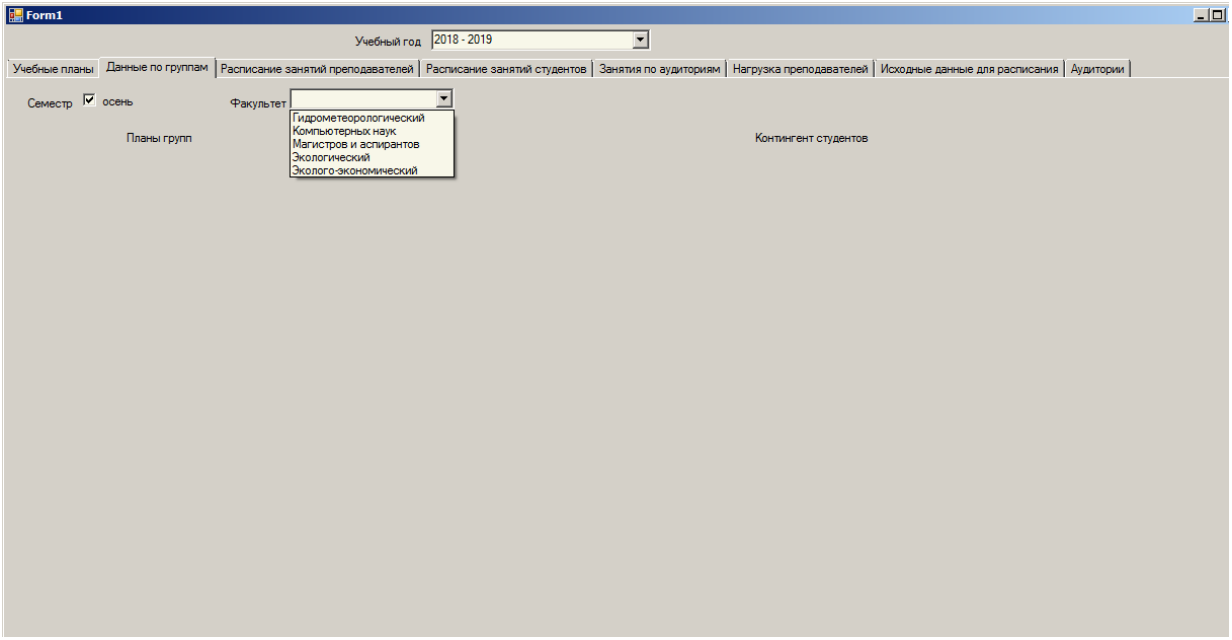

Рисунок 7.5 – Вибір факультету для виведення інформації по групах

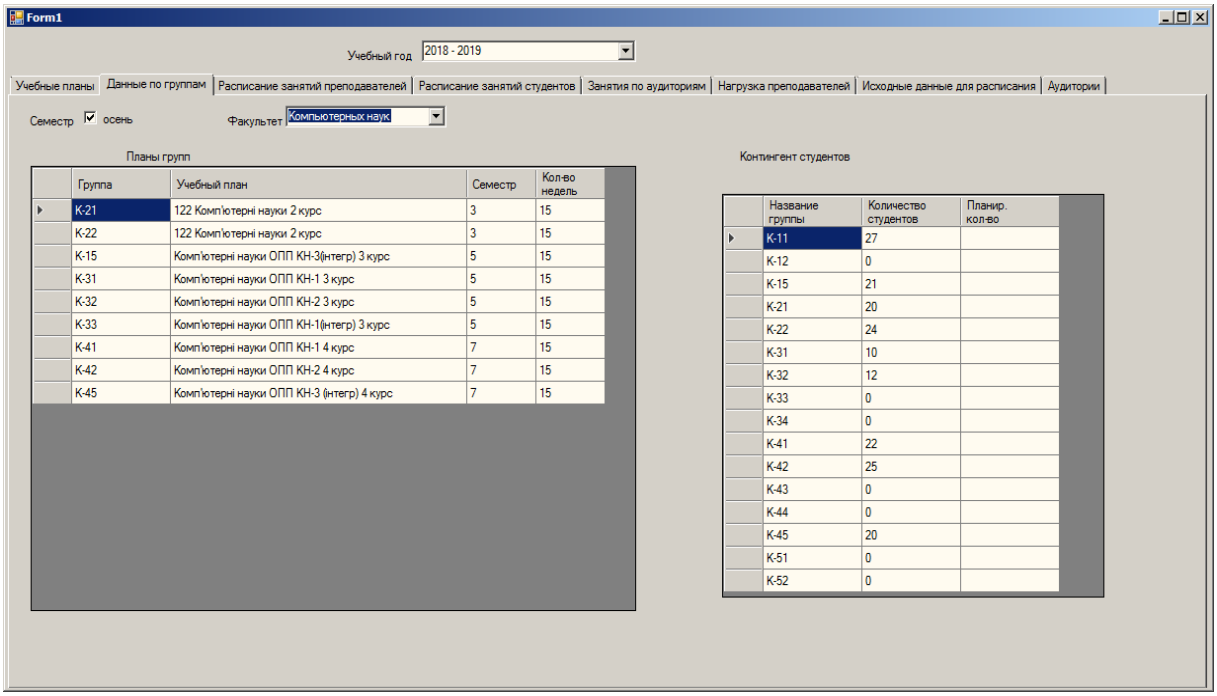

Рисунок 7.6 – Перегляд контингенту студентів та інформації про прикріплення груп до навчальних планів

Третя вкладка «Расписание работы преподавателей» дозволяє переглянути розклад занять усіх викладачів університету.

Для цього потрібно спочатку вибрати кафедру, де працює викладач (рис. 7.7).

Потім зі списку викладачів вибраної кафедри вибрати викладача, розклад якого бажано переглянути (рис. 7.8).

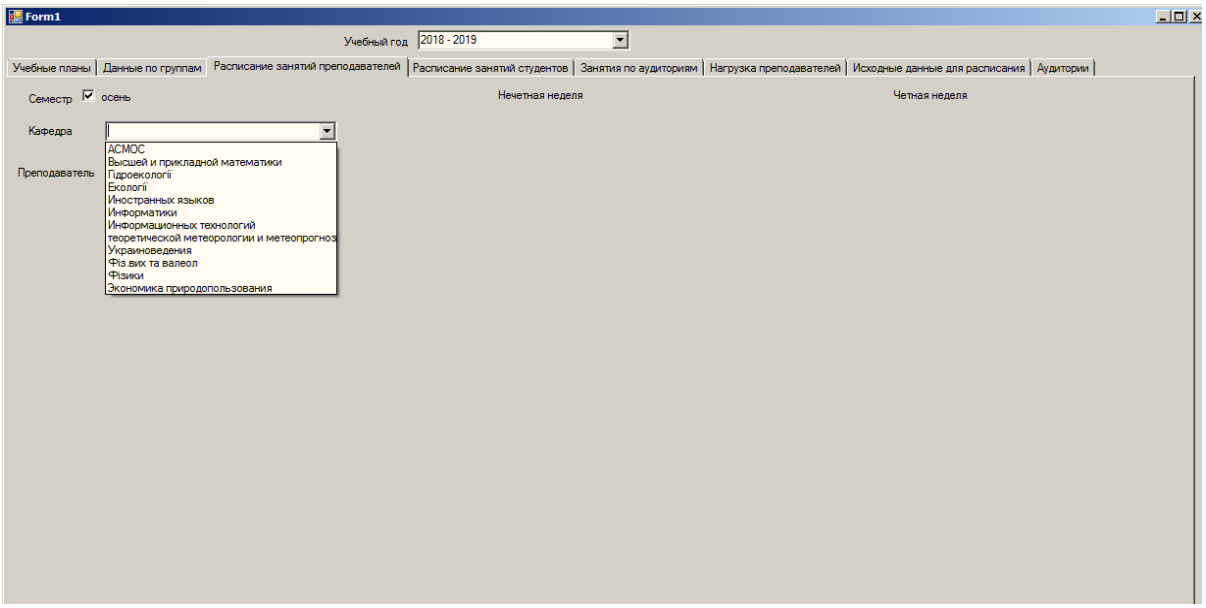

Рисунок 7.7 – Вибір кафедри для отримання списку викладачів

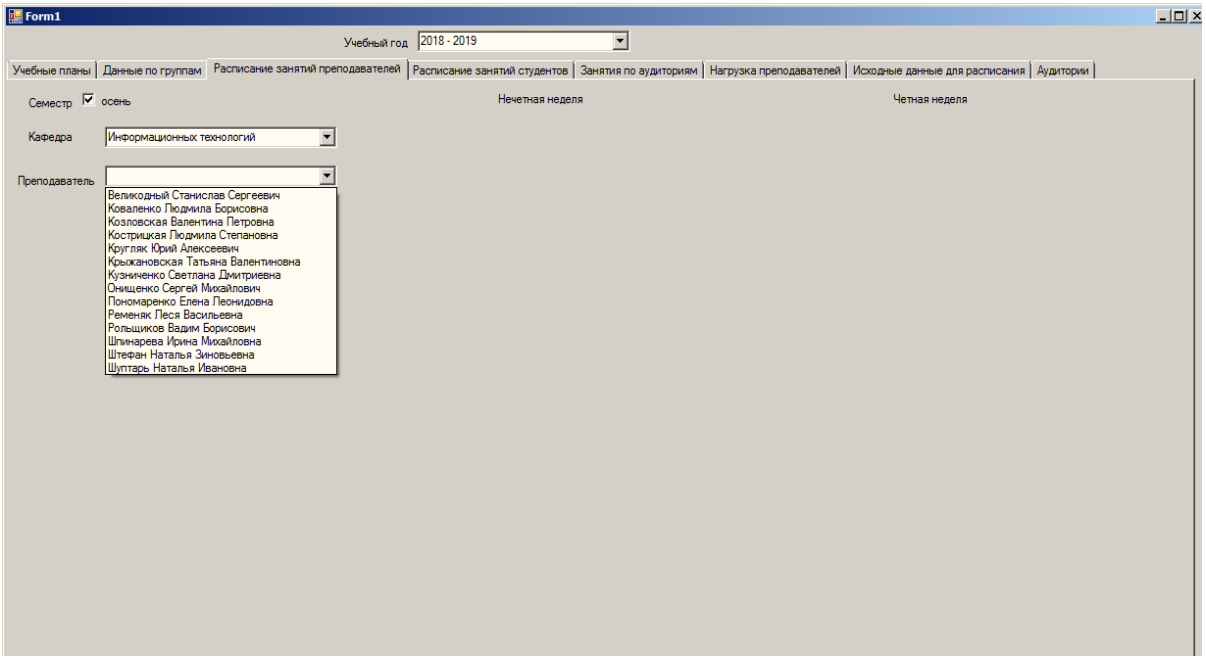

Рисунок 7.8 – Вибір викладача для перегляду розкладу занять

Після цього на формі з’являються табличні компоненти з розкладом занять вибраного викладача (рис. 7.9).

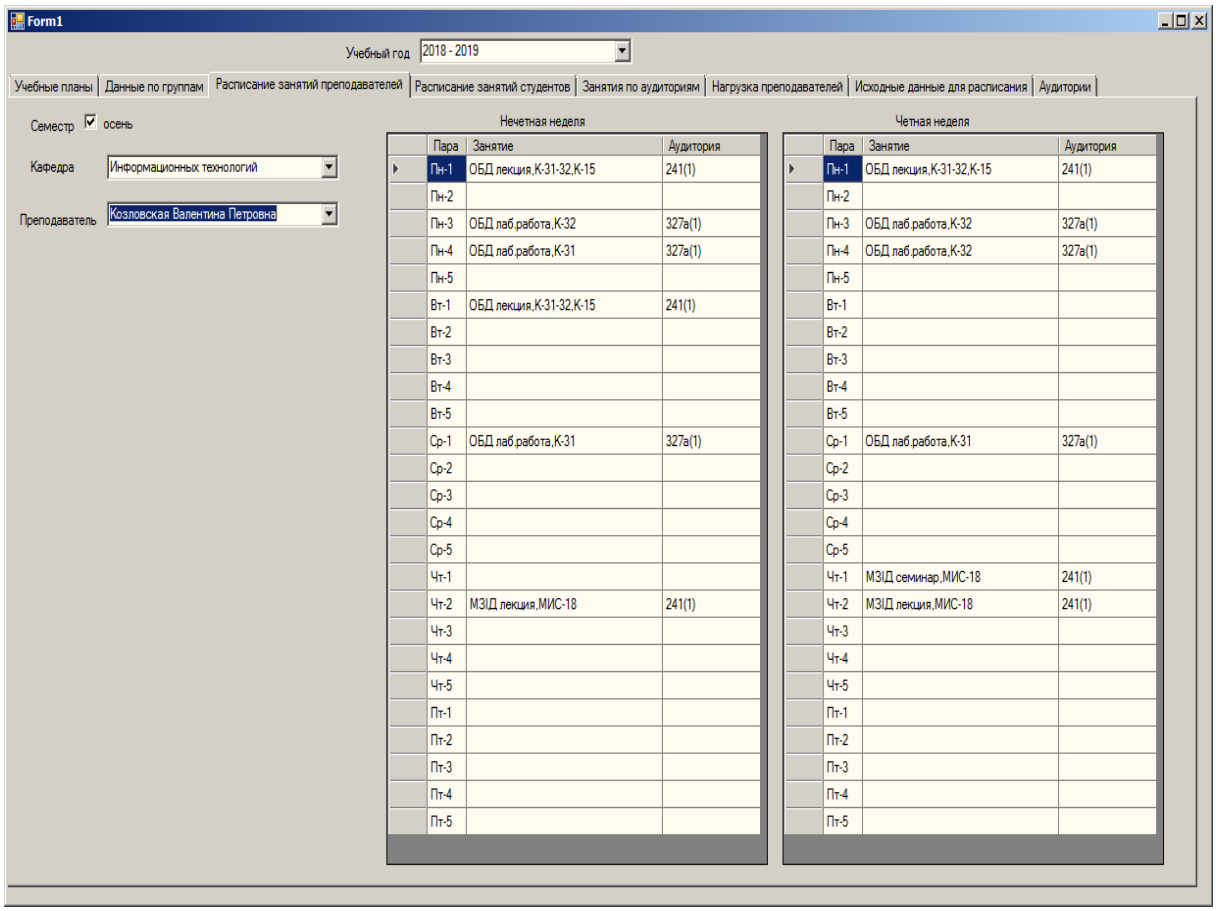

Рисунок 7.9 – Вибір викладача для перегляду розкладу занять

Розклад занять виводиться у два табличні компоненти: ліворуч виводиться розклад занять на непарному тижні (у тижнях з непарними номерами), праворуч – розклад занять на парному тижні.

Розклад занять виводиться в таблиці з трьома стовпчиками. У першому стовпчику виводиться скорочено назва дня тижня та номер навчальної пари, наприклад, «Пн-1».

У другому стовпчику виводиться скорочена назва навчальної дисципліни, вид заняття та назва академічної групи або потоку, наприклад, «МЗІД лекція, МІС-18» або «ОБД лаб.работа, К-15а».

У третьому стовпчику виводиться номер аудиторії, де проводиться заняття з записом у дужках номеру навчального корпусу, наприклад, «327а(1)».

Вільні від занять навчальні пари виводяться порожніми. У розкладі виводяться по 5 пар у 5 робочих днів тижня.

Четверта вкладка «Расписание занятий студентов» дозволяє переглянути розклад занять усіх академічних груп університету.

Для цього потрібно спочатку вибрати факультет, до якого відноситься група (рис. 7.10).

Потім зі списку груп вибрати групу, розклад якої бажано переглянути (рис. 7.11).

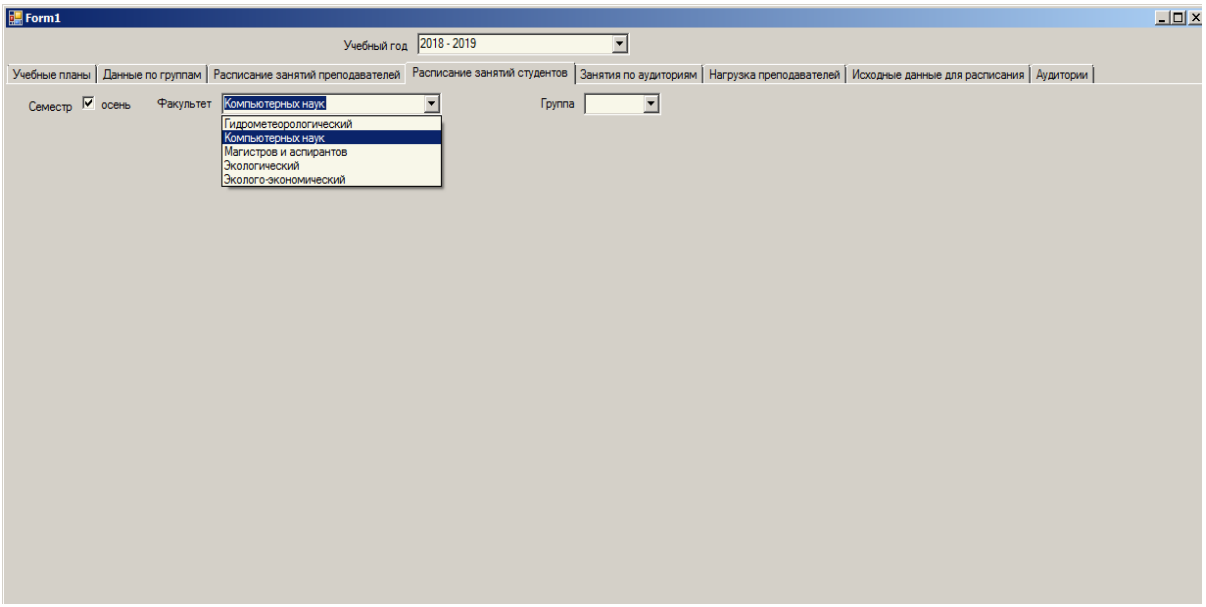

Рисунок 7.10 – Вибір факультету, розклад занять груп якого переглядається

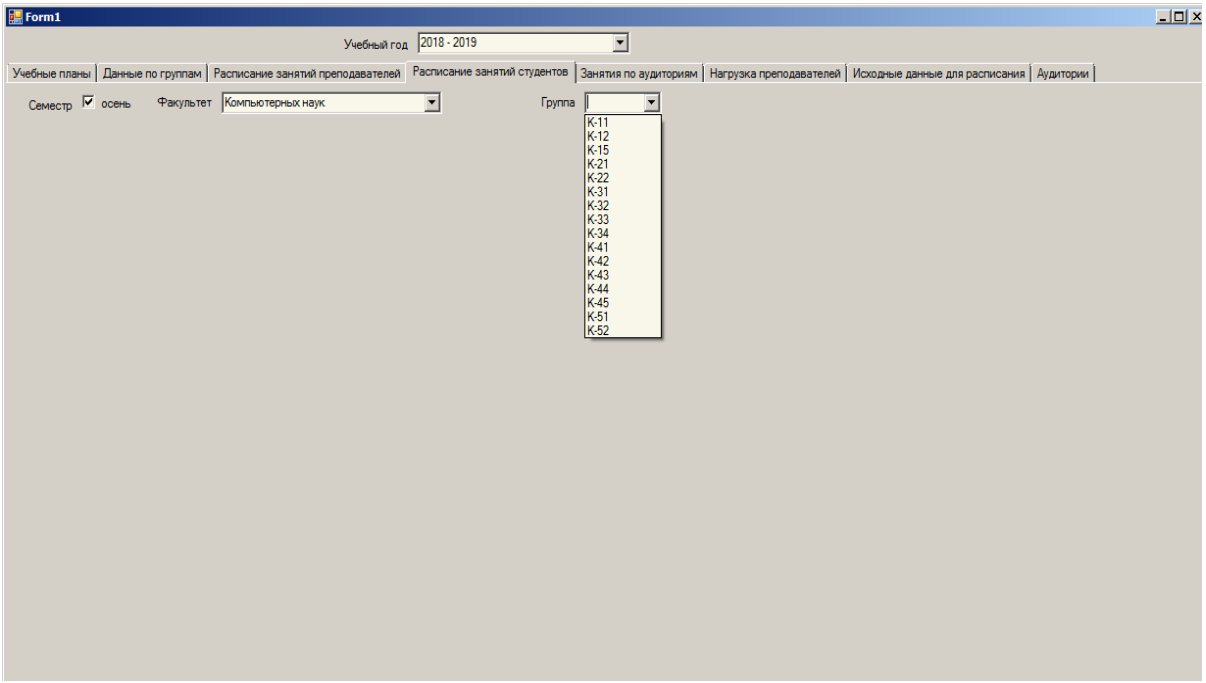

Рисунок 7.11 – Вибір групи для перегляду розкладу занять

Після цього на формі з’являються табличні компоненти з розкладом занять вибраної групи (рис. 7.9).

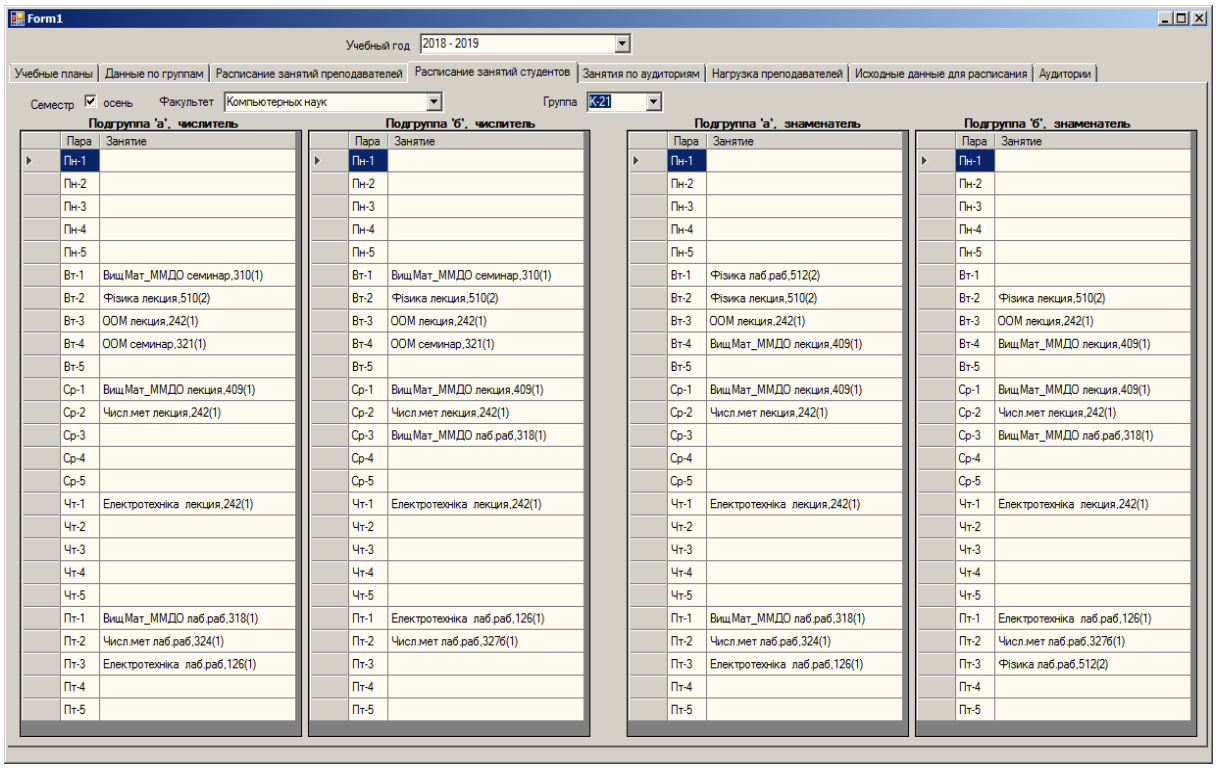

Рисунок 7.12 – Перегляд розкладу занять вибраної групи

Якщо у вибраної групи немає жодного заняття, що проводиться з поділом на підгрупи, то розклад занять виводиться у дві таблиці, як у викладачів: розклад занять на непарному та на парному тижнях семестру.

Якщо хоча б на одному тижні у групи є заняття з поділом на підгрупи, розклад занять виводиться у чотири табличні компоненти: розклад занять підгрупи «а» на непарному тижні, розклад занять підгрупи «б» на непарному тижні, розклад занять підгрупи «а» на парному тижні, розклад занять підгрупи «б» на парному тижні.

Розклад занять виводиться в таблиці з двома стовпчиками. У першому стовпчику виводиться скорочено назва дня тижня та номер навчальної пари, наприклад, «Пн-1».

У другому стовпчику виводиться скорочена назва навчальної дисципліни, вид заняття та номер аудиторії проведення заняття, наприклад, «ООМ семінар, 321(1)».

Вільні від занять навчальні пари виводяться порожніми. У розкладі виводяться також по 5 пар у 5 робочих днів тижня.

П’ята вкладка «Занятия по аудиториям» дозволяє переглянути розклад занять у аудиторіях навчальних корпусів університету.

Для цього потрібно спочатку вибрати корпус, до якого відноситься аудиторія (рис. 7.13).

Потім зі списку аудиторій вибраного корпусу вибрати ту аудиторію, завантаження якої бажано переглянути (рис. 7.14).

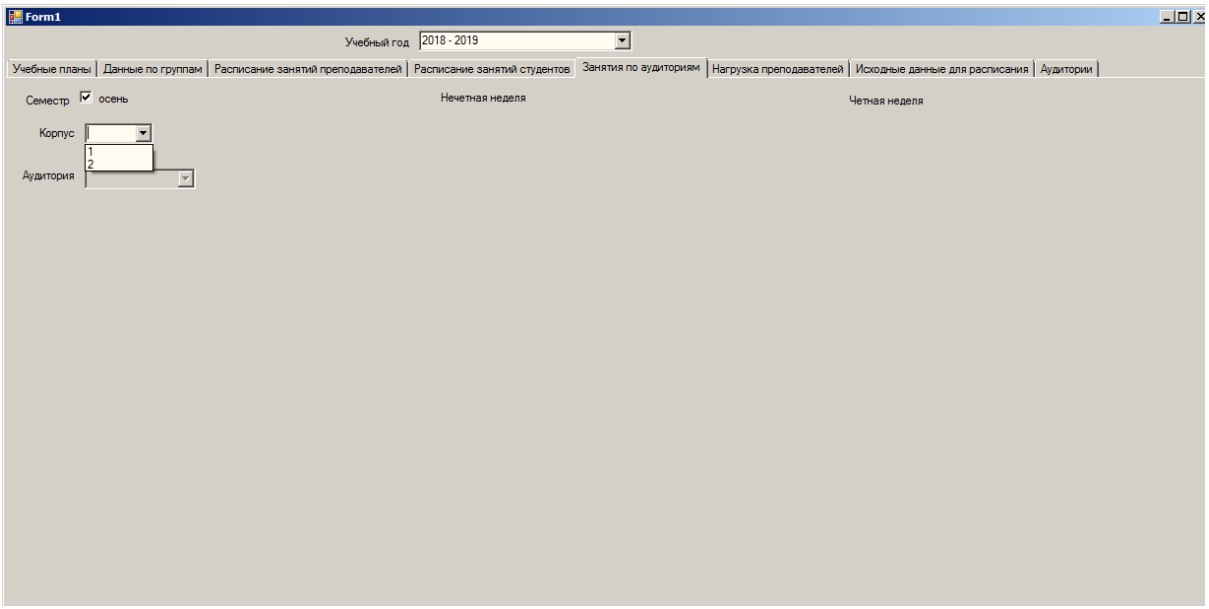

Рисунок 7.13 – Вибір корпусу для вибору аудиторії

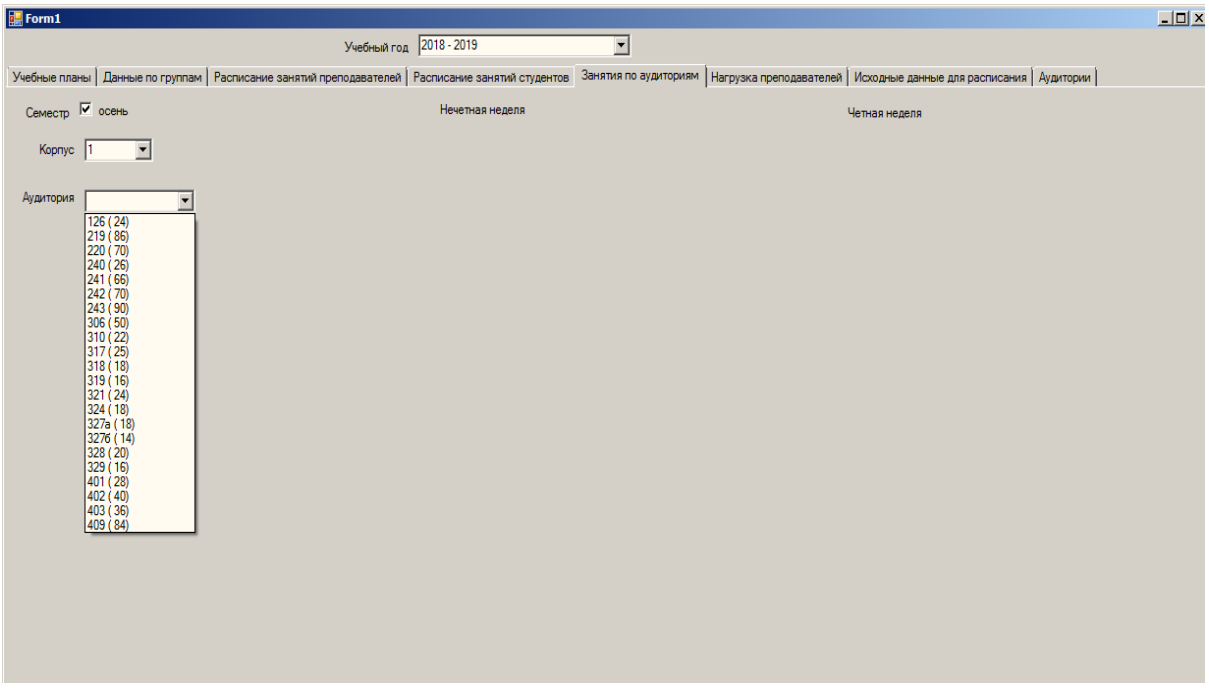

Рисунок 7.14 – Вибір аудиторії для перегляду її завантаження

Після цього можна переглянути завантаження вибраної аудиторії заняттями (рис. 7.15)

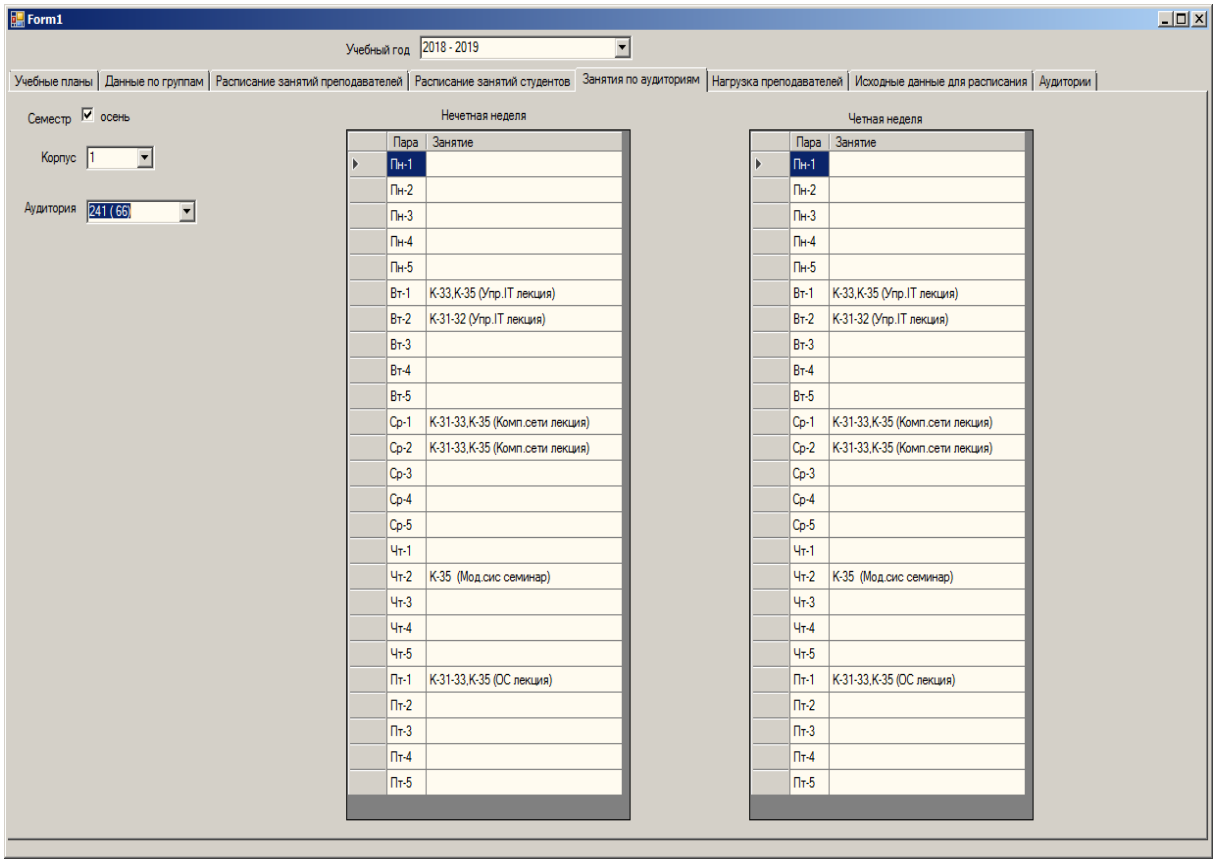

Рисунок 7.15 – Перегляд завантаження заняттями вибраної аудиторії вибраного корпусу

Завантаження аудиторії заняттями – розклад занять в аудиторії – виводиться у дві таблиці, як у викладачів: розклад занять на непарному та на парному тижнях семестру.

Розклад занять виводиться в таблиці з двома стовпчиками. У першому стовпчику виводиться скорочено назва дня тижня та номер навчальної пари, наприклад, «Пн-1».

У другому стовпчику виводиться назва академічної групи або потоку, для яких в аудиторії проводиться заняття, а також у дужках виводиться скорочена назва навчальної дисципліни та вид заняття, наприклад, «К-31-33, К-35 (Комп. сети лекция)».

Вільні від занять навчальні пари виводяться порожніми. У розкладі виводяться також по 5 пар у 5 робочих днів тижня.

Наступна вкладка дозволяє переглянути аудиторне навчальне навантаження викладачів університету у вибраному навчальному році.

Спочатку також треба вибрати кафедру, де працює викладач. Потім зі списку викладачів потрібно вибрати викладача, навантаження якого бажано переглянути (рис. 7.16). Потім потрібно вибрати, заплановане або виконане навантаження викладача треба переглянути. Потім на формі програми стає видимим аудиторне навантаження на вибраний навчальний рік для вибраного викладача (рис. 7.17 – 7.18).

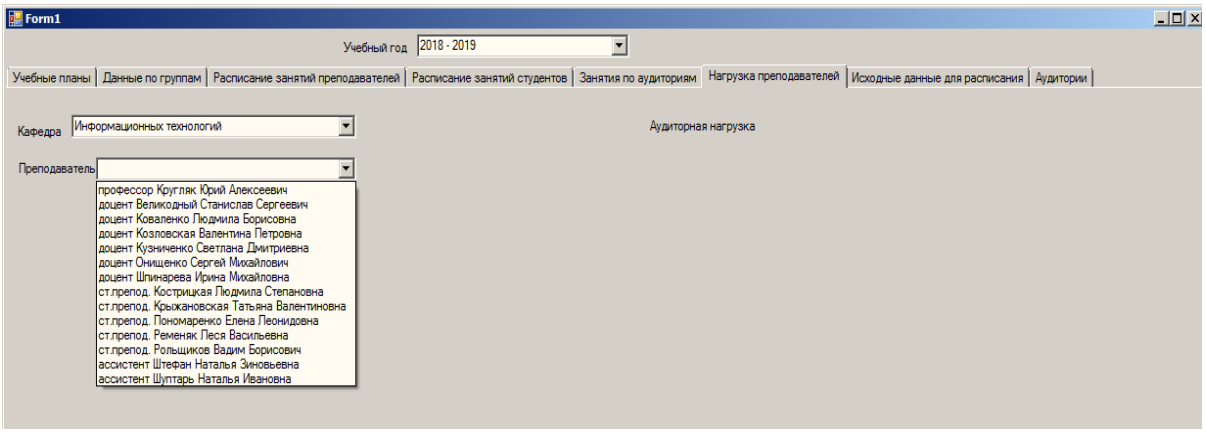

Рисунок 7.16 – Вибір кафедра та викладача для перегляду навантаження

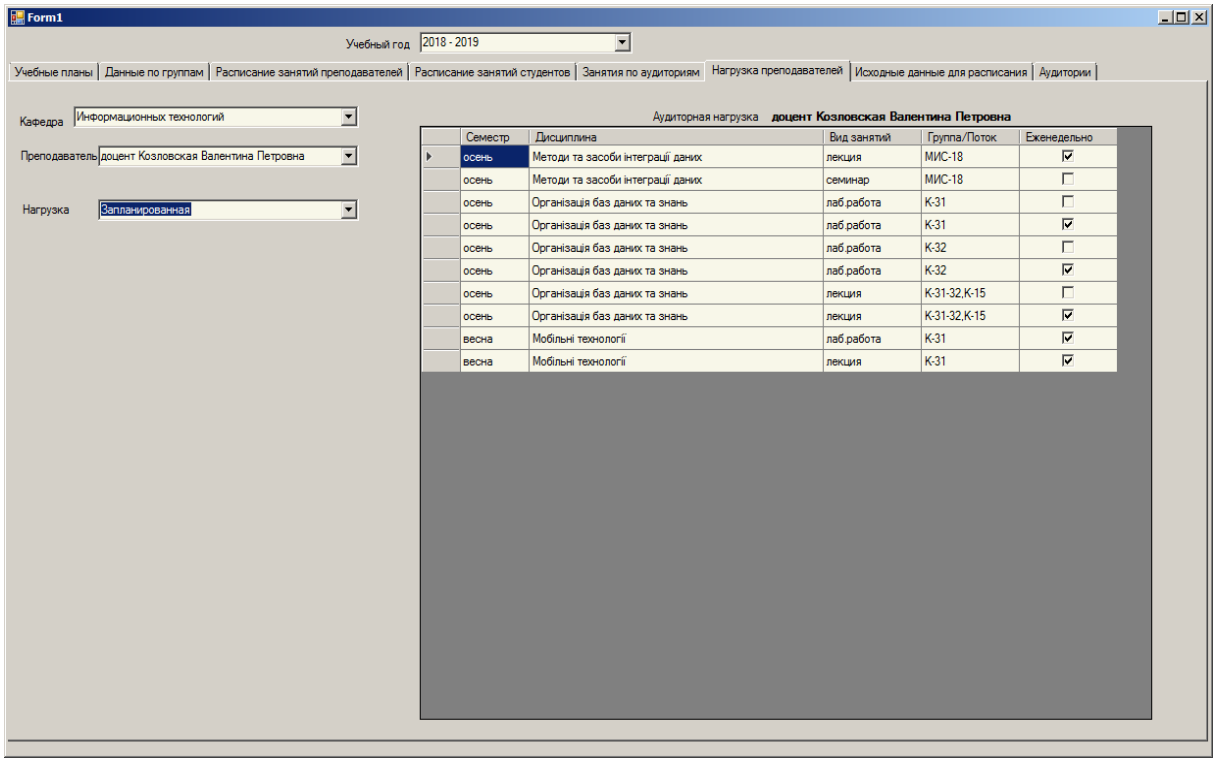

Рисунок 7.17 – Заплановане аудиторне навантаження вибраного викладача

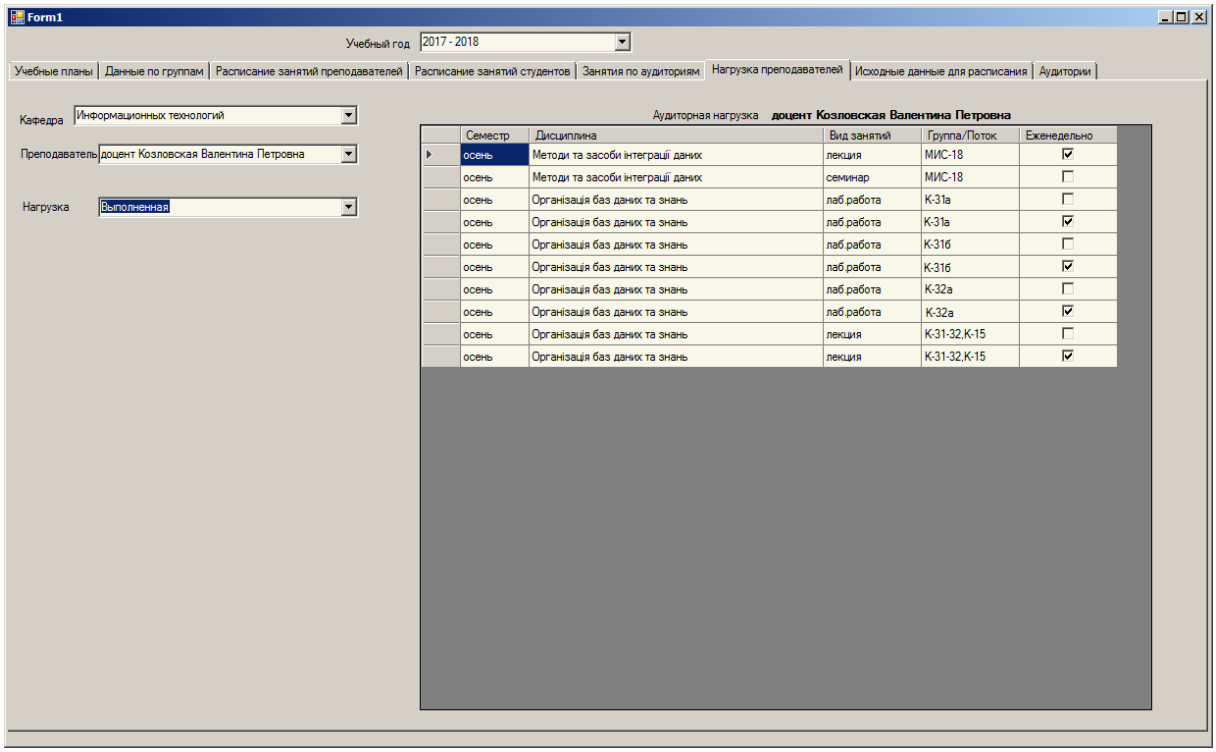

Рисунок 7.18 – Виконане аудиторне навантаження вибраного викладача

Остання вкладка програми «Аудитории» дозволяє переглядати наявність аудиторій у навчальних корпусах та вносити зміни у їх список або місткість (рис. 7.19).

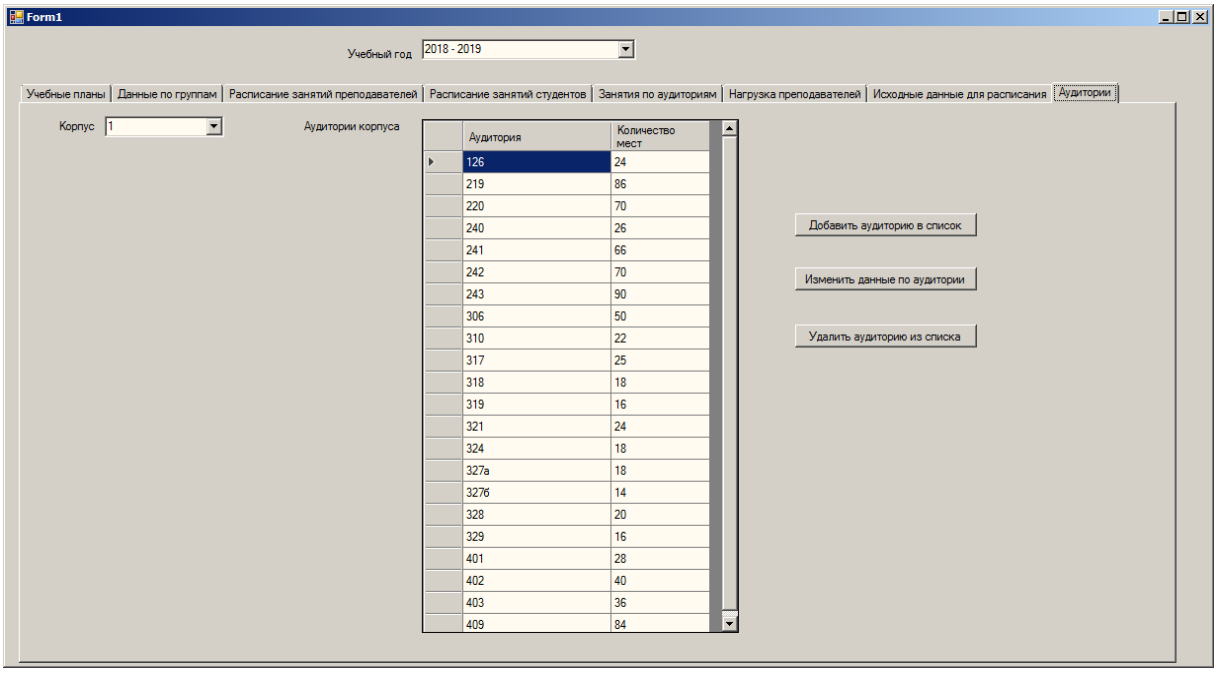

Рисунок 7.19 – Перегляд списку аудиторій вибраного корпусу

Якщо список аудиторій корпусу у БД не повний, можна додати аудиторію у БД, натиснувши кнопку «Добавить аудиторию в список». Після цього на формі стають видимими компоненти для додавання аудиторії (рис.7.20).

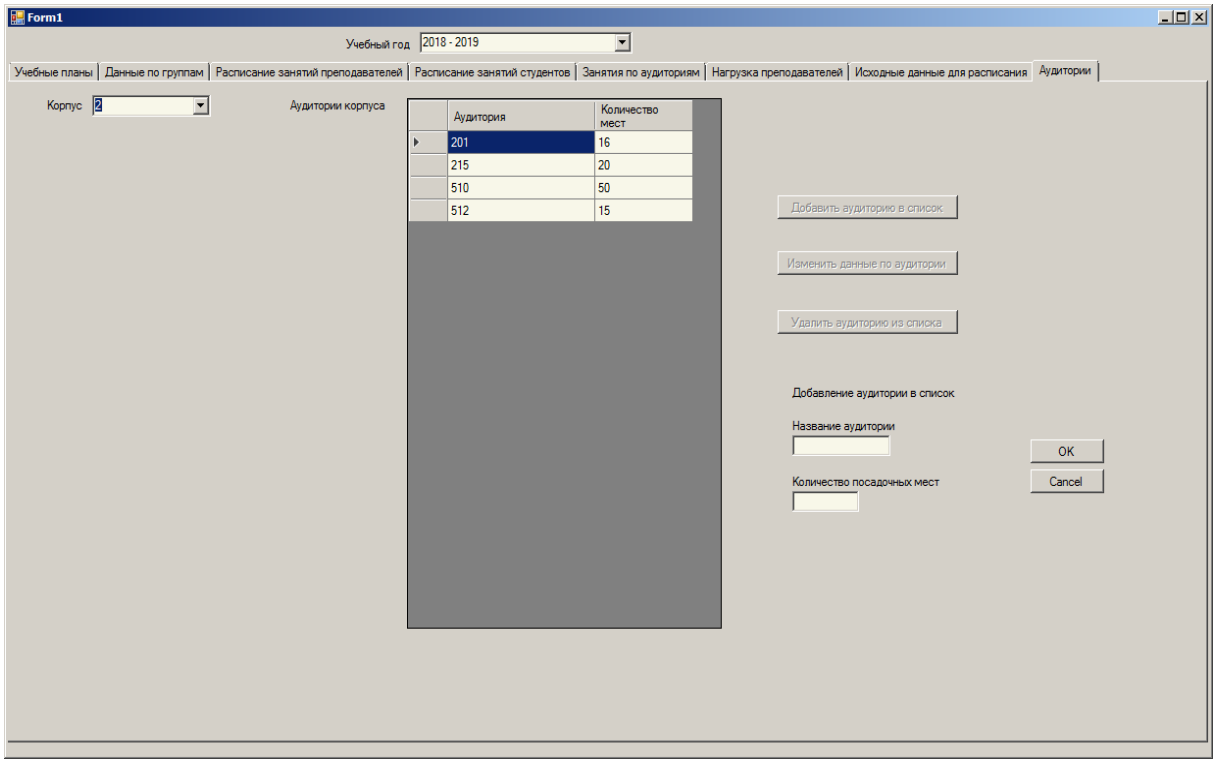

Рисунок 7.20 – Можливість додавання аудиторії у список аудиторій вибраного корпусу

Також можна видалити аудиторію із списку навчальних аудиторій корпусу, якщо натиснути кнопку «Удалить аудиторию из списка». Програма запросить підтвердження необхідності видалення запису з таблиці бази даних (рис. 7.21).

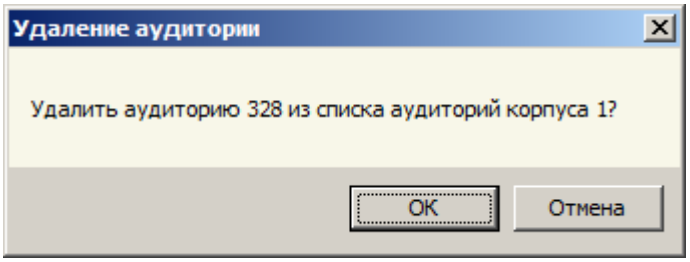

Рисунок 7.21 – Запит про необхідності видалення запису з БД

Якщо користувач натисне «OK», вказана аудиторія видалиться із списку аудиторій у БД.

Змінити назву або місткість аудиторії можна, якщо натиснути кнопку «Изменить данные по аудитории» на формі програми. Тоді стають видимими приховані компоненти форми для внесення змін у запис в таблиці бази даних (рис. 7.22).

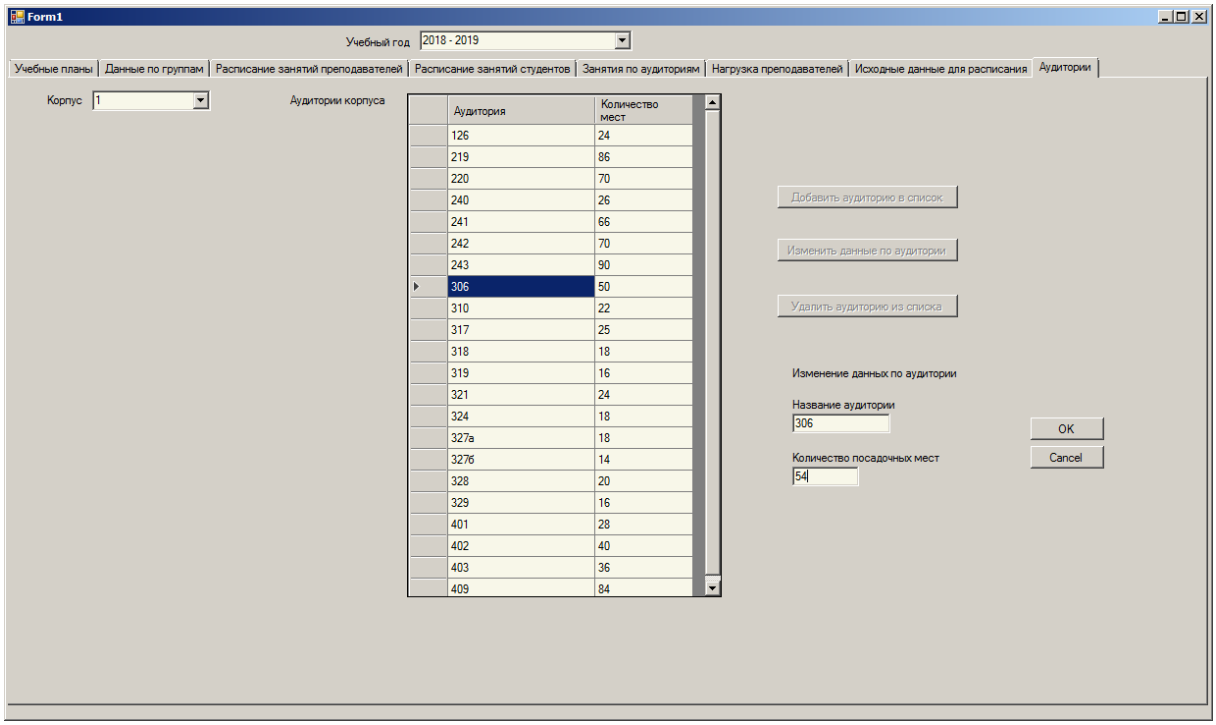

Рисунок 7.22 – Можливість внесення змін у запис в таблиці аудиторій БД

Місткість аудиторій перевіряється, коли для заняття призначається список допустимих аудиторій. Якщо місткість аудиторії виявиться меншою, ніж кількість студентів у групі або потоку, для яких проводиться заняття, буде видане попередження про неможливість призначити задану аудиторію для проведення заданого заняття.

ВИСНОВКИ

Дана магістерська робота присвячена моделюванню та розробці підсистеми «Навчальний відділ» інформаційної системи «Навчальний процес університету». Розробка єдиної інформаційної системи, що описує навчальний процес в університеті від створення навчальних планів до проведення поточних та підсумкових контролюючих заходів дозволить скоротити потік документів між підрозділами університету.

Навчальний відділ університету вносить вагомий внесок у навчальний процес. Він складає розклад занять, контролює їх проведення, контролює планування викладачами навчального навантаження та звітність про виконання цього навантаження. Програма виду «Автоматизоване робоче місце працівника навчального відділу» допоможе у роботі працівникам цього відділу та спростить обмін документами між цим відділом, деканатами факультетів, кафедрами та іншими підрозділами університету.

Результатами виконання даної магістерської роботи є:

- розроблена концептуальна модель підсистеми «Навчальний відділ» у складі ІС «Навчальний процес університету»;
- відповідно до моделі змінена схема бази даних база даних: додано 4 нових таблиці;
- розроблено нове серверне програмне забезпечення для підсистеми «Навчальний відділ»;
- розроблене серверне програмне забезпечення включає в себе 6 збережених процедур і 12 уявлень на мові T-SQL для СУБД MS SQL Server 2008;
- розроблено клієнтське застосування «Автоматизоване робоче місце працівника навчального відділу» у вигляді офісного додатку до бази даних інформаційної системи «Навчальний процес університету»;
- клієнтське застосування дозволяє працівнику навчального відділу переглядати розклад занять студентів, викладачів, завантаженість навчальних аудиторій, заплановане та виконане аудиторне завантаження викладача, визначати готовність БД системи до складання розкладу занять, оновлювати дані в БД щодо аудиторного фонду університету.

Програма «Автоматизоване робоче місце працівника навчального відділу» розроблялась як частина інформаційної системи «Навчальний

процес університету», тому не вимагає введення великих масивів вихідних даних.

ПЕРЕЛІК ПОСИЛАНЬ

1. Куликов Г. Г., Никулина Н. О., Речкалов А. В. Управление проектами на основе системного моделирования: Учебное пособие. Уфа: УГАТУ, 2009. – 171 с.
2. Вендров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем. М: «Финансы и статистика», 1998. – 98 с.
3. Томас Конноли, Каролин Бегг. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3-е издание.: Пер. с англ. М.: Издательский дом «Вильямс», 2003. – 1440 с.
4. Дейт К. Дж.. Введение в системы баз данных, 6-е издание: Пер. с англ. – К.; М.; СПб: Издательский дом «Вильямс», 2000. – 848 с.
5. Сравнение SQL баз данных. http://webarty.net/databases/sravnenie-sql-baz-dannyh
6. Мамаев Е.В. Microsoft SQL Server 2000. – СПб.: БХВ-Петербург, 2004. – 1260 с.
7. Дэвидсон Л. Проектирование баз данных на SQL Server 2000 / Л.Дэвидсон; пер. с англ. – М.: БИНОМ. Лаборатория знаний, 2003. –680 с.
8. Роберт Виейра. Программирование баз данных Microsoft SQL Server 2005. Базовый курс. М.: «Диалектика», 2007. – с. 832.
9. Введение в MS SQL Server и T-SQL https://metanit.com/sql/sqlserver/1.1.php
10. Интегрированная среда разработки Visual Studio. URL: https://docs.microsoft.com/ru-ru/visualstudio/ide/visual-studio-ide?view
11. Описание среды разработки Microsoft Visual Studio. URL: https://studbooks.net/2258619/informatika/opisanie_sredy_razrabotki_microsoft_visual_studio
12. Знакомство с Visual Studio. URL: https://habr.com/sandbox/79099.
13. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. RL: https://metanit.com/sharp/tutorial/
14. Руководство по программированию на C#. URL: https://docs.microsoft.com/ru-ru/previous-versions/67ef8sbd
15. Руководство по программированию на C#. URL: https://docs.microsoft.com/ru-ru/previous-versions/67ef8sbd

Д О Д А Т К И

ДОДАТОК А
ЛОГІЧНА СХЕМА БАЗИ ДАНИХ

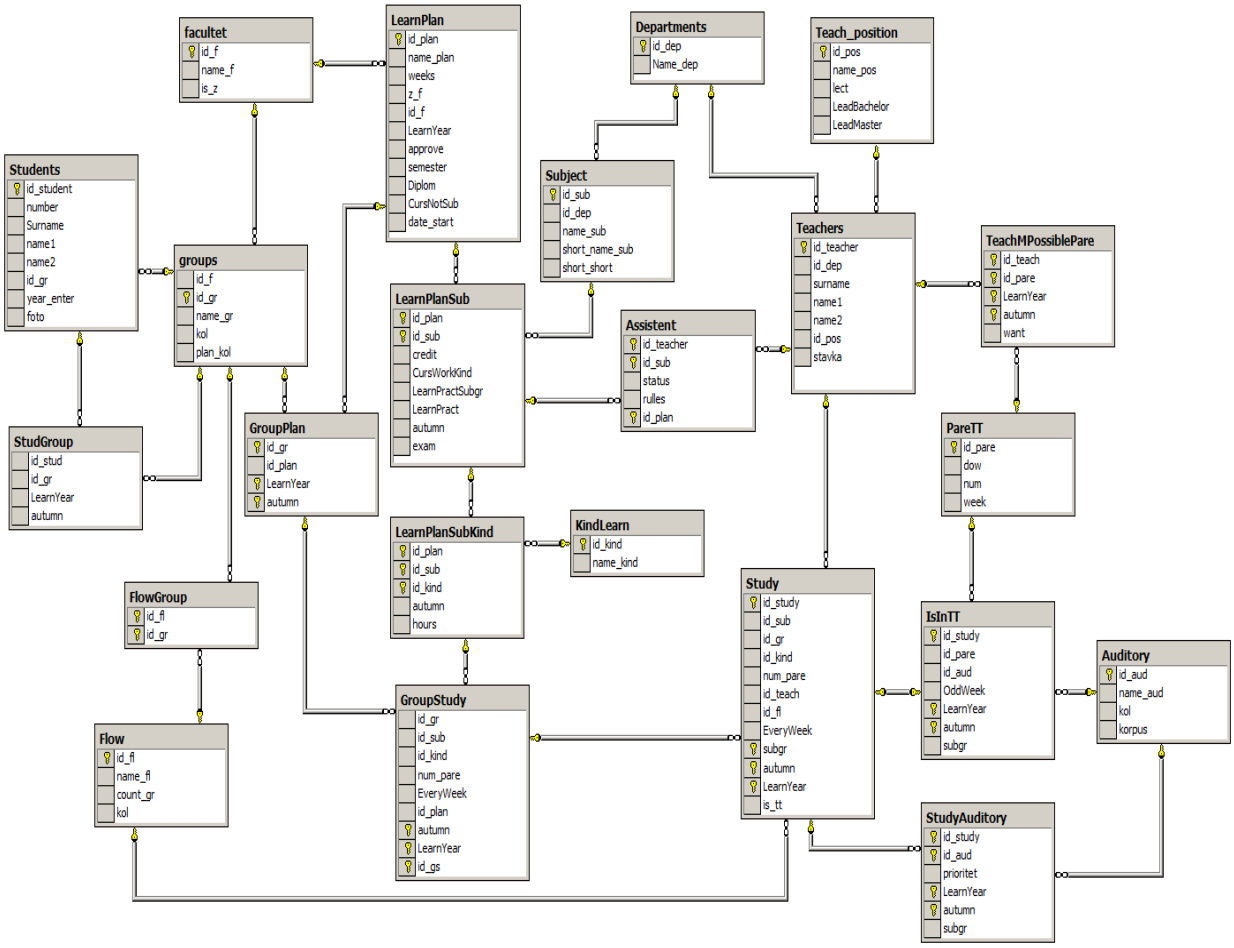

Рисунок А.1 – Підсхема «Навчальний відділ» ІС «Навчальний процес університету»

ДОДАТОК Б
ФІЗИЧНА СХЕМА БАЗИ ДАНИХ

```
CREATE TABLE [StudyPlan] (
	[id_study] [int] NOT NULL ,
	[id_sub] [smallint] NOT NULL ,
	[id_gr] [smallint] NOT NULL ,
	[id_kind] [tinyint] NOT NULL ,
	[num_pare] [tinyint] NOT NULL ,
	[id_teach] [smallint] NULL ,
	[id_fl] [smallint] NULL ,
	[EveryWeek] [bit] NULL ,
	[sub_gr] [bit] NULL ,
	[subgr] [tinyint] NULL ,
	[autumn] [bit] NOT NULL ,
	[LearnYear] [smallint] NOT NULL ,
	[is_tt] [bit] NULL ,
	CONSTRAINT [PK__StuPlan] PRIMARY KEY  CLUSTERED
	(
		[id_study],
		[LearnYear],
		[autumn]
	)  ON [PRIMARY] ,
	CONSTRAINT [FK__StPlan__id_fl] FOREIGN KEY
	(
		[id_fl]
	) REFERENCES [Flow] ([id_fl]),
	CONSTRAINT [FK__StPlan__id_teach] FOREIGN KEY
	(
		[id_teach]
	) REFERENCES [Teachers] ( [id_teacher] )
) ON [PRIMARY]
GO

CREATE TABLE [SubFlow] (
	[id_sub] [smallint] NOT NULL ,
	[id_plan] [smallint] NOT NULL ,
	[id_fl] [smallint] NULL ,
	 PRIMARY KEY  CLUSTERED
	(
		[id_plan],
		[id_sub]
	)  ON [PRIMARY] ,
	 FOREIGN KEY
	(
		[id_plan],
		[id_sub]
	) REFERENCES [LearnPlanSub] (
		[id_plan],
		[id_sub]
	),
	 FOREIGN KEY
	(
		[id_fl]
	) REFERENCES [Flow] (
		[id_fl]
	)
) ON [PRIMARY]
GO
```

```
CREATE TABLE [SubSubgr] (
	[id_sub] [smallint] NOT NULL ,
	[id_plan] [smallint] NOT NULL ,
	[id_gr] [smallint] NOT NULL REFERENCES groups,
	[id_kind] [tinyint] NOT NULL  REFERENCES KindLearn,
	 PRIMARY KEY  CLUSTERED
	(
		[id_plan],
		[id_sub],
		[id_gr],
		[id_kind]
	)  ON [PRIMARY] ,
	 FOREIGN KEY
	(
		[id_plan],
		[id_sub]
	) REFERENCES [LearnPlanSub] (
		[id_plan],
		[id_sub]
	)
)
GO

CREATE TABLE MayPareTeach (
	id_teach smallint NOT NULL REFERENCES Teachers,
	id_pare smallint NOT NULL REFERENCES PareTT,
	desire bit not null,
	 PRIMARY KEY (id_teach, id_pare)
)
GO
```

ДОДАТОК В
СЕРВЕРНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

```
create proc AuditorAdd
@name_aud varchar(6),
@kol tinyint,
@korpus tinyint,
@k smallint output,
@msg varchar(100) output
as
begin
        set @k=(select k=count(*) from Auditory
                where name_aud=@name_aud and korpus = @korpus)
        if @k=0
                insert Auditory(name_aud, korpus,kol )values(@name_aud,@korpus,@kol  )
        else
   set @msg='В '+Str(@korpus,2)+'-м корпусе уже есть аудитория названием  '+@name_aud
end
GO
create proc AuditorEdit
@id_aud smallint,
@name_aud varchar(6),
@kol tinyint,
@korpus tinyint,
@k smallint output,
@msg varchar(100) output
as
begin
        set @k=(select k=count(*) from Auditory
                where name_aud=@name_aud and korpus = @korpus and id_aud!=@id_aud )
        if @k=0
        begin
                set @msg='OK'
        update Auditory set name_aud=@name_aud, korpus=@korpus,kol=@kol where  id_aud=@id_aud
        end
        else
set @msg='В '+Str(@korpus,2)+'-м корпусе уже есть другая аудитория названием '+@name_aud
end
GO
CREATE  procedure AuditorDel
  @id_aud smallint ,
  @m tinyint,
  @k tinyint output,
  @msg varchar (150) output
as
begin
        if @m>0
        begin
        delete IsInTT where id_aud = @id_aud  delete StudyAuditory where id_aud = @id_aud
        delete Auditory where id_aud = @id_aud
        set @k=0  set @msg = 'OK'
        end
        else
        begin
                set @k = (select k=count(*) from IsInTT where id_aud = @id_aud)
                if @k = 0
                        delete Auditory where id_aud = @id_aud
                else
                        set @msg='В эту аудиторию назначено '+str(@k,3)+' занятий.'+char(13)+char(10)+'Все
равно удалять аудиторию вместе с занятиями в расписании?'
        end
end
GO
```

ДОДАТОК Г
ВИХІДНИЙ КОД КЛІЄНТСЬКОГО ЗАСТОСУВАННЯ

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

namespace Learning
{
    public partial class Form1 : Form
    {
        short LearnYear = 2017, id_teach = 66, id_dep = 1, id_plan = 69, id_gr = 67,
id_sub = 65, id_f = 1, id_aud = 14;
        byte korpus = 1;
        string name = "";
        int aut=1;
        bool prevE = false, prevD = false, autumn = true;
        private System.Data.OleDb.OleDbCommand addAudCmd;
        private System.Data.OleDb.OleDbCommand editAudCmd;
        private System.Data.OleDb.OleDbCommand delAudCmd; Sys-
tem.Data.OleDb.OleDbConnection Connection;
        public Form1()
        {
            InitializeComponent();
            Connection = this.auditoryTableAdapter.Connection;
            fillFacComboBox();
            fillDepComboBox();
            loadComboBox.SelectedIndex = 0;
            corpusComboBox.SelectedIndex = 0;
            this.auditoryTableAdapter.Fill(this.dataSet1.Auditory, korpus);
            this.lP_viewTableAdapter.Fill(this.dataSet1.LP_view, id_plan);
            this.view_GroupPlanTableAdapter.Fill(this.dataSet1.View_GroupPlan, id_f,aut
);
            this.groupsTableAdapter.Fill(this.dataSet1.groups, id_f);
            this.teachTT_week1TableAdapter.Fill(this.dataSet1.TeachTT_week1, id_teach,
autumn);
            this.teachTT_week0TableAdapter.Fill(this.dataSet1.TeachTT_week0, id_teach,
autumn);
            this.viewTT_10TableAdapter.Fill(this.dataSet2.ViewTT_10, id_gr);
            this.viewTT_11TableAdapter.Fill(this.dataSet2.ViewTT_11, id_gr);
            this.viewTT_00TableAdapter.Fill(this.dataSet2.ViewTT_00, id_gr);
            this.viewTT_01TableAdapter.Fill(this.dataSet2.ViewTT_01, id_gr);
            this.view_TT0_audTableAdapter.Fill(this.dataSet2.View_TT0_aud, id_aud);
            this.view_TT1_audTableAdapter1.Fill(this.dataSet2.View_TT1_aud, id_aud);
            this.dataTable2TableAdapter.Fill(this.dataSet1.DataTable2, id_teach);
        }

        private void init_addAudCmd()
        {
            addAudCmd = new System.Data.OleDb.OleDbCommand();
            addAudCmd.Connection = this.auditoryTableAdapter.Connection;
            addAudCmd.CommandText = "AuditAdd";
            addAudCmd.CommandType = System.Data.CommandType.StoredProcedure;
            addAudCmd.Parameters.Add("@id_aud", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
```

```
            addAudCmd.Parameters.Add("@name_aud", System.Data.OleDb.OleDbType.VarChar,
60);
            addAudCmd.Parameters.Add("@Korpus", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt);
            addAudCmd.Parameters.Add("@kol", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt);
            addAudCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            addAudCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }

        private void init_EditThCmd()
        {
            editAudCmd = new System.Data.OleDb.OleDbCommand();
            editAudCmd.Connection = this.auditoryTableAdapter.Connection;
            editAudCmd.CommandText = "AuditEdit";
            editAudCmd.CommandType = System.Data.CommandType.StoredProcedure;
            editAudCmd.Parameters.Add("@id_aud", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt);
            editAudCmd.Parameters.Add("@name_aud", System.Data.OleDb.OleDbType.VarChar,
60);
            editAudCmd.Parameters.Add("@Korpus", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt);
            editAudCmd.Parameters.Add("@kol", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt);
            editAudCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            editAudCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }
        private void init_DelThCmd()
        {
            delAudCmd = new System.Data.OleDb.OleDbCommand();
            delAudCmd.Connection = this.auditoryTableAdapter.Connection;
            delAudCmd.CommandText = "AuditDel";
            delAudCmd.CommandType = System.Data.CommandType.StoredProcedure;
            delAudCmd.Parameters.Add("@id_aud", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt);
            delAudCmd.Parameters.Add("@k", Sys-
tem.Data.OleDb.OleDbType.UnsignedTinyInt).Direction = Sys-
tem.Data.ParameterDirection.InputOutput;
            delAudCmd.Parameters.Add("@msg", System.Data.OleDb.OleDbType.VarChar,
100).Direction = System.Data.ParameterDirection.InputOutput;
        }

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

            }
            try
            {
                System.Data.OleDb.OleDbDataReader reader = depCmd.ExecuteReader();
                List_Item li;
```

```
                depCom-
boBox.Items.Clear();depComboBox2.Items.Clear();depComboBox3.Items.Clear();
                int i = 0;
                while (reader.Read())
                {
                    li = new List_Item(reader.GetInt16(0), reader.GetString(1));
                    depCom-
boBox.Items.Add(li);depComboBox2.Items.Add(li);depComboBox3.Items.Add(li);
                    ++i;
                }
                reader.Close();
                depComboBox.SelectedIndex = -1;depComboBox2.SelectedIndex = -
1;depComboBox3.SelectedIndex = -1;

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

        private void fillFacComboBox()
        {
            Connection = this.auditoryTableAdapter.Connection;
            System.Data.OleDb.OleDbCommand facCmd = new Sys-
tem.Data.OleDb.OleDbCommand("Select id_f, name_f from facultet where is_z=0 order by
name_f", Connection);
            System.Data.ConnectionState previousConnectionState = Connection.State;
            if (((Connection.State & System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open))
            {
                Connection.Open();

            }
            try
            {
                System.Data.OleDb.OleDbDataReader reader = facCmd.ExecuteReader();
                List_Item li;
                facComboBox.Items.Clear();facComboBox2.Items.Clear();
                facComboBox3.Items.Clear(); facComboBox4.Items.Clear(); com-
boBox2.Items.Clear();
                int i = 0;
                while (reader.Read())
                {
                    li = new List_Item(reader.GetInt16(0), reader.GetString(1));
                    facComboBox.Items.Add(li);facComboBox2.Items.Add(li);
                    facComboBox3.Items.Add(li); facComboBox4.Items.Add(li); com-
boBox2.Items.Add(li);
                    ++i;
                }
                reader.Close();
                facComboBox.SelectedIndex = -1;facComboBox2.SelectedIndex = -1;
                facComboBox3.SelectedIndex = -1; facComboBox4.SelectedIndex = -1; com-
boBox2.SelectedIndex = -1;

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

        private void fillTeachComboBox1(short id_dep)
```

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

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

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

        private void fillTeachComboBox(short id_dep)
        {
            Connection = this.auditoryTableAdapter.Connection;
            System.Data.OleDb.OleDbCommand teachCmd = new Sys-
tem.Data.OleDb.OleDbCommand("Select id_teacher, prep=name_pos+' '+surname+' '+name1+'
'+name2 from Teachers t join teach_position p on t.id_pos=p.id_pos where id_dep="

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

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

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

        private void fillGroupComboBox(short id_f)
        {
            Connection = this.auditoryTableAdapter.Connection;
            System.Data.OleDb.OleDbCommand groupCmd = new Sys-
tem.Data.OleDb.OleDbCommand("Select id_gr, name_gr from groups where id_f=" +
id_f.ToString() + " order by name_gr", Connection);
            System.Data.ConnectionState previousConnectionState = Connection.State;
            if (((Connection.State & System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open))
            {
                Connection.Open();

            }
            try
            {
                System.Data.OleDb.OleDbDataReader reader = groupCmd.ExecuteReader();
                List_Item li;
                groupComboBox.Items.Clear();comboBox1.Items.Clear();
                int i = 0;
                while (reader.Read())
                {
                    li = new List_Item(reader.GetInt16(0), reader.GetString(1));
                    groupComboBox.Items.Add(li);comboBox1.Items.Add(li);
                    ++i;
                }
                reader.Close();
                groupComboBox.SelectedIndex = -1;comboBox1.SelectedIndex = -1;

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

        private void fillAudComboBox(byte corpus)
        {
            Connection = this.auditoryTableAdapter.Connection;
            System.Data.OleDb.OleDbCommand audCmd = new Sys-
tem.Data.OleDb.OleDbCommand("Select id_aud, name_a= name_aud+' ('+STR(kol,3)+')' from Au-
ditory where Korpus=" + corpus.ToString() + " order by name_aud", Connection);
            System.Data.ConnectionState previousConnectionState = Connection.State;
            if (((Connection.State & System.Data.ConnectionState.Open)
                        != System.Data.ConnectionState.Open))
            {
                Connection.Open();

            }
            try
            {
                System.Data.OleDb.OleDbDataReader reader = audCmd.ExecuteReader();
                List_Item li;
                audComboBox.Items.Clear();
                int i = 0;
                while (reader.Read())
                {
```

```
                li = new List_Item(reader.GetInt16(0), reader.GetString(1));
                audComboBox.Items.Add(li);
                ++i;
            }
            reader.Close();
            audComboBox.SelectedIndex = -1;

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

    private void Form1_Load(object sender, EventArgs e)
    {
        this.view_TT0_audTableAdapter.Fill(this.dataSet2.View_TT0_aud, id_aud);
        this.view_TT1_audTableAdapter1.Fill(this.dataSet2.View_TT1_aud, id_aud);
        this.viewTT_10TableAdapter.Fill(this.dataSet2.ViewTT_10, id_gr);
        this.view_TT1_audTableAdapter.Fill(this.dataSet1.View_TT1_aud);
        this.teachTT_week0TableAdapter.Fill(this.dataSet1.TeachTT_week0, id_teach,
autumn);
        this.teachTT_week1TableAdapter.Fill(this.dataSet1.TeachTT_week1, id_teach,
autumn);
        this.groupsTableAdapter.Fill(this.dataSet1.groups, id_f);
        this.view_GroupPlanTableAdapter.Fill(this.dataSet1.View_GroupPlan, id_f,
aut);
        this.lP_viewTableAdapter.Fill(this.dataSet1.LP_view, id_plan);
        this.auditoryTableAdapter.Fill(this.dataSet1.Auditory, korpus);
        this.learnPlanTableAdapter.Fill(this.dataSet1.LearnPlan, id_f);
        this.dataTable2TableAdapter.Fill(this.dataSet1.DataTable2, id_teach);
        this.viewGroupAudLoadTableAdapter.Fill(this.dataSet1.ViewGroupAudLoad, id_gr,
autumn);

    }

    private void comboBox4_SelectedIndexChanged(object sender, EventArgs e)
    {
        tabControl1.Visible = true;
    }

    private void facComboBox_SelectedIndexChanged(object sender, EventArgs e)
    {
        dataGridView2.Visible = true;
        id_f = (byte)(((List_Item)(facComboBox.SelectedItem)).id);

        this.learnPlanTableAdapter.Fill(this.dataSet1.LearnPlan, id_f);
        int k = dataGridView2.RowCount;
        if (k > 0)
        {
            id_plan = (short)(dataGridView2.Rows[0].Cells[5].Value);
            this.lP_viewTableAdapter.Fill(this.dataSet1.LP_view, id_plan);
            dataGridView3.Visible = true;
        }
    }

    private void corpusComboBox_SelectedIndexChanged(object sender, EventArgs e)
    {
        korpus = Convert.ToByte(corpusComboBox.SelectedItem.ToString());
        this.auditoryTableAdapter.Fill(this.dataSet1.Auditory, korpus);
    }


    private void dataGridView2_CellEnter(object sender, DataGridViewCellEventArgs e)
```

```
        {
            id_plan = (short)(dataGridView2.Rows[e.RowIndex].Cells[5].Value);
            this.lP_viewTableAdapter.Fill(this.dataSet1.LP_view, id_plan);
        }

        private void facComboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_f = (byte)(((List_Item)(facComboBox2.SelectedItem)).id);
            this.view_GroupPlanTableAdapter.Fill(this.dataSet1.View_GroupPlan, id_f,
aut);
            this.groupsTableAdapter.Fill(this.dataSet1.groups, id_f);
            dataGridView5.Visible = true; dataGridView4.Visible = true;
        }

        private void checkBox2_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox2.Checked) aut = 1; else aut = 0;
        }

        private void depComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_dep = (short)(((List_Item)(depComboBox.SelectedItem)).id);
            fillTeachComboBox1(id_dep);
            teachComboBox1.Enabled = true;
        }

        private void teachComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_teach = (short)(((List_Item)(teachComboBox1.SelectedItem)).id);
            if (id_teach > 0)
            {
                this.teachTT_week1TableAdapter.Fill(this.dataSet1.TeachTT_week1,
id_teach, autumn);
                this.teachTT_week0TableAdapter.Fill(this.dataSet1.TeachTT_week0,
id_teach, autumn);
                dataGridView7.Visible = true; dataGridView6.Visible = true;
            }
        }

        private void checkBox3_CheckedChanged(object sender, EventArgs e)
        {
            autumn = checkBox3.Checked;
        }

        private void facComboBox3_SelectedIndexChanged(object sender, EventArgs e)
        {
            short id_f = (short)(((List_Item)(facComboBox3.SelectedItem)).id);
            fillGroupComboBox(id_f);
        }

        private void groupComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_gr = (short)(((List_Item)(groupComboBox.SelectedItem)).id);
            this.viewTT_10TableAdapter.Fill(this.dataSet2.ViewTT_10, id_gr);
            this.viewTT_11TableAdapter.Fill(this.dataSet2.ViewTT_11, id_gr);
            this.viewTT_00TableAdapter.Fill(this.dataSet2.ViewTT_00, id_gr);
            this.viewTT_01TableAdapter.Fill(this.dataSet2.ViewTT_01, id_gr);
            dataGridView11.Visible = true;dataGridView10.Visible = true;
            dataGridView8.Visible = true;dataGridView9.Visible = true;
        }

        private void coprComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            korpus = Convert.ToByte(coprComboBox.SelectedItem.ToString());
            fillAudComboBox(korpus); audComboBox.Enabled = true;
```

```
        }

        private void audComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_aud = (short)(((List_Item)(audComboBox.SelectedItem)).id);
            this.view_TT0_audTableAdapter.Fill(this.dataSet2.View_TT0_aud, id_aud);
            this.view_TT1_audTableAdapter1.Fill(this.dataSet2.View_TT1_aud, id_aud);
            dataGridView12.Visible = true; dataGridView13.Visible = true;
        }

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

        private void delButton_Click(object sender, EventArgs e)
        {
            byte korpus = (byte)(dataGridView1.CurrentRow.Cells[3].Value);
            string corpus = korpus.ToString();
            string name_aud = (string)(dataGridView1.CurrentRow.Cells[0].Value);
            string s = "Удалить аудиторию " + name_aud + " из списка аудиторий корпуса "+
corpus+"?";
            if (MessageBox.Show(s, "Удаление аудитории", MessageBoxButtons.OKCancel) ==
System.Windows.Forms.DialogResult.OK)
            {
                delAudCmd.Parameters["@id_aud"].Value =
(short)(dataGridView1.CurrentRow.Cells[2].Value);
                delAudCmd.Parameters["@korpus"].Value = korpus;
                delAudCmd.Parameters["@m"].Value = 0;
                delAudCmd.Parameters["@k"].Value = System.DBNull.Value;
                delAudCmd.Parameters["@msg"].Value = System.DBNull.Value;
                System.Data.ConnectionState previousConnectionState = delAud-
Cmd.Connection.State;
                if (((delAudCmd.Connection.State & System.Data.ConnectionState.Open)
                            != System.Data.ConnectionState.Open))
                {
                    delAudCmd.Connection.Open();
                }
                try
                {
                    delAudCmd.ExecuteNonQuery();
                    if ((byte)(delAudCmd.Parameters["@k"].Value) > 0)
                    {
                        if (MessageBox.Show((string)(delAudCmd.Parameters["@msg"].Value),
"Удаление из связанных таблиц", MessageBoxButtons.OKCancel) == Sys-
tem.Windows.Forms.DialogResult.OK)
                        {
                            delAudCmd.Parameters["@m"].Value = 1;
                            delAudCmd.ExecuteNonQuery();
                        }
                    }
                }
                finally
                {
                    if ((previousConnectionState == System.Data.ConnectionState.Closed))
                        delAudCmd.Connection.Close();
                }
                this.view_TT0_audTableAdapter.Fill(this.dataSet2.View_TT0_aud, id_aud);
            }
        }

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

```
            short id_aud = (short)(dataGridView1.CurrentRow.Cells[2].Value);
        }

        private void checkBox4_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox4.Checked) autumn = true; else autumn = false;
        }

        private void loadComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            bool load = true;
        }

        private void teachComboBox_SelectedIndexChanged_1(object sender, EventArgs e)
        {
            id_teach = (short)(((List_Item)(teachComboBox.SelectedItem)).id);
            teachLabel.Text = teachComboBox.SelectedItem.ToString();
            this.dataTable2TableAdapter.Fill(this.dataSet1.DataTable2, id_teach);
            dataGridView14.Visible = true;teachLabel.Visible = true;
        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
            short id_f = (short)(((List_Item)(comboBox2.SelectedItem)).id);
            fillGroupComboBox(id_f);
        }

        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            id_gr = (short)(((List_Item)(comboBox1.SelectedItem)).id);
            this.viewGroupAudLoadTableAdapter.Fill(this.dataSet1.ViewGroupAudLoad, 67,
true);
        }
    }

    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;
        }
    }
}
```