МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет Магістерської та аспірантської підготовки
Кафедра інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка серверного програмного забезпечення для програми автоматизованого складання розкладу занять. Розробка моделі та алгоритму урахування в програмі занять з різною кількістю годин на тиждень. »

Виконав студент 2 курсу групи МК- 61
спеціальності 122 Комп'ютерні науки та інформаційні технології
Літовенко Юлія Ігорівна

Керівник: Козловська В.П. к.ф.-м.н., доцент

Консультант _____

Рецензент : Худенко Н.П., к.т.н., доцент

Одеса 2018

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ЗВО – заклад вищої освіти.

ОПП – освітньо-професійна програма підготовки фахівців

СУБД – система управління базами даних

FPA – ім’я відношення, що описує вільні навчальні пари в аудиторіях (Free Pare Auditory).

FPG – ім’я відношення, що описує вільні навчальні пари академічних груп (Free Pare Group).

FPT – ім’я відношення, що описує вільні навчальні пари викладачів (Free Pare Teacher).

PA – ім’я відношення, що описує допустимі для проведення занять навчальні пари в аудиторіях (Pare Auditory).

PG – ім’я відношення, що описує допустимі для постановки занять у розклад навчальні пари академічних груп (Pare Group).

PT – ім’я відношення, що описує допустимі для постановки занять у розклад навчальні пари викладачів (Pare Teacher).

SA – ім’я відношення, що описує допустимі для проведення заняття аудиторії (Study Auditory).

STT – ім’я відношення, що описує заняття, що вже поставлені у розклад (Study in TimeTable).

ВСТУП

Задача складання розкладів є предметом наукових досліджень з середини минулого століття. Сфера їх застосування включає різні сфери людської діяльності, такі як: транспортні перевезення, масове обслуговування, промисловість, освіта і так далі. Практика висуває безліч задач, які неможливо ефективно вирішити шляхом повного перебору. Існує навіть наука з назвою «Теорія розкладу», проте, вона дозволяє отримати чітке рішення для обмеженого круга задач, наприклад, для задачі комівояжера. Задача складання розкладу занять до них не відноситься, вона є обчислювально складною, тому що має багато критеріїв, які повинні враховуватись.

Ця задача виникає з року в рік у будь-якій навчальній установі. Складання розкладу навчальних занять є одним з найважливіших завдань управління навчальним процесом. У зв’язку з цим проблема автоматизації складання розкладу занять в освітніх системах масового навчання як і раніше залишається однією з актуальних проблем організації навчального процесу. Дійсно, від того наскільки «вдало» складений розклад занять залежать якість навчання, економічна ефективність навчання, комфортність навчання студентів і роботи професорсько-викладацького складу і багато що інше.

Задача складання розкладів відноситься до задач цілочисельного програмування, і вже не перший рік для її вирішення застосовують комп’ютерні програми. На ринку програмного забезпечення пропонується багато програм автоматизованого складання розкладу занять. Але всі ці програми не мають відкритого коду, тому неможливо визначити, за яким алгоритмом знаходиться рішення задачі.

В Інтернеті пропонується велика кількість таких програм, при цьому усі вони платні, і коштують немало, особливо ті, які відзначаються критиками як досить хороші. Дешевші програмні продукти мають істотні недоліки: незручний інтерфейс, урахування далеко не всіх вимог і побажань, незадовільний кінцевий розклад, проблематичність зручного коригування розкладу.

У сучасних скрутних економічних умовах далеко не всі університети можуть дозволити собі витрати на купівлю дорогих програм для складання розкладу занять. Як наслідок, у більшості університетів нині не використовуються програми автоматичного складання розкладу занять, а розклад складається вручну диспетчерами навчального відділу університету.

Отже, хоч задача складання розкладу занять студентів в університеті не може вважатися новою, вона досі не втратила своєї актуальності.

При розробці алгоритму автоматизованого складання розкладу занять в вищому навчальному закладі виникає ціла низка складних задач. Однією з цих задач є урахування занять з різною кількістю годин на тиждень. Стандартним заняттям вважається заняття одного виду (лекція, практичне, лабораторне) з однієї дисципліни впродовж двох академічних годин. Але у навчальних планах ЗВО присутні навчальні дисципліни, для яких призначено як одну годину якогось виду занять, так і N годин, де $N > 2$.

Дослідження методів вирішення задачі складання розкладу занять при наявності занять з довільною кількістю годин на тиждень є завданням цієї магістерської роботи.

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. Розробка окремої програми складання розкладу занять в університеті потребує введення великої кількості вихідних даних. Використання подібної програми у якості складової частини інформаційної системи «Навчальний процес університету» дозволяє використовувати в програмі дані, які отримані на інших стадіях навчального процесу: навчальні плани факультетів та контингенти студентів, розподіл аудиторного навантаження між викладачами кафедр, тощо.

При проектуванні ІС «Навчальний процес університету» необхідно розглянути всі підсистеми, які беруть участь у даному процесі.

1.1 Об’єкти та суб’єкти навчального процесу університету

У закладі вищої освіти студент набуває освіту за допомогою аудиторних занять та самостійної роботи, при цьому в останні роки суттєво збільшилась частка часу, який відводиться під самостійну роботу студента. Для можливості проведення аудиторних занять навчальний відділ ЗВО складає розклад занять на кожний поточний семестр. Основою для складання розкладу занять є навчальні плани за спеціальностями, контингент студентів, та заплановане аудиторне навантаження викладачів.

Навчальні плани за спеціальностями розробляються уповноваженими комісіями, уточнюються деканатами факультетів і затверджуються методичним відділом університету. Основою навчальних планів за спеціальності є освітньо-професійна програма підготовки фахівців (ОПП) та інші нормативні документи.

Навчальні дисципліни розподіляються між кафедрами університету. На кафедрі за кожною дисципліною закріплюється провідний викладач, який відповідає за результати навчання студентів з даної дисципліни. Провідний викладач проводить аудиторні заняття з дисципліни сам або за допомогою асистентів. Розподіл занять між викладачами кафедри складає навчальне навантаження викладача. Частина навчального навантаження викладача – аудиторне навантаження – є одною з складових для складання розкладу занять у ЗВО.

Крім списків розподілу занять між викладачами кафедри надають у навчальний відділ також списки допустимих аудиторій для проведення

кожного заняття. Для кожного заняття можна вказувати або перелік допустимих аудиторій, або навчальний корпус, в якому повинно проводитись заняття. Якщо вказаний лише навчальний корпус, то з усіх неспеціалізованих аудиторій цього корпусу для проведення заняття повинні вибиратись аудиторії, найбільш допустимі з урахуванням кількості студентів групи та кількості посадкових місць аудиторії.

1.2 Загальні вимоги до задачі складання розкладу занять

Для розробки програми слід спочатку спроектувати базу даних для неї. Необхідність використання бази даних витікає з великого обсягу даних, які використовуються при рішенні задачі складання розкладу.

Як правило, передбачається, що задача складання розкладу занять в університеті повинна вирішуватися методами лінійного програмування, що допускає використання вихідних даних, або записаних у вигляді файлів, або отриманих вибіркою з бази даних. Проте, через великий обсяг вихідних даних рекомендується вихідні дані для задачі складання розладу занять в вищому навчальному закладі отримувати з бази даних.

Задача складання розкладу занять добре формулюється в термінах теорії множин, теорії відношень і реляційної алгебри. Саме теорія відношень і реляційна алгебра є теоретичною основою реляційних систем. Отже, логічно припустити, що реляційні СУБД можуть використовуватися не лише як сховище вихідних даних для задачі, що розглядається, але їх програмні засоби можуть використовуватися для знаходження рішення цієї задачі.

1.3 Використання у навчальних планах дисциплін з різною кількістю годин на тиждень

У закладах вищої освіти одиницею часового відрізку для проведення заняття визначена академічна навчальна пара – дві академічні години занять поспіль. Простіше за все складати розклад занять у випадках, коли з будь-якого виду занять – лекції, семінари, лабораторні заняття, – в навчальному плані передбачено 2 години занять на тиждень. Але цей випадок є скоріш виключенням, ніж правилом у наш час.

В останні роки частка аудиторного навантаження студентів зменшилася на користь самостійної підготовки. Цей чинник став причиною появи в навчальних планах багатьох дисциплін, у яких заплановано лише 1

годину занять по якомусь виду. З іншого боку, зменшення кількості навчальних дисциплін, які можуть вивчати студенти протягом семестру, призвело до об’єднання в навчальних планах в одну дисципліну декількох дисциплін. Наприклад, раніше були можливі у навчальних планах декілька навчальних дисциплін, які за суттю вивчали різні розділи вищої математики. Деякі з цих дисциплін вивчались послідовно, інші могли читатись паралельно в одному семестрі. Тепер у для спеціальностей, де вища математика не є фаховою дисципліною, у навчальних планах може стояти одна дисципліна з назвою «Вища математика» з різною кількістю годин на тиждень.

Тобто звичайним для навчальних планів є випадок дисциплін з N годинами на тиждень, де N, як правило, має будь-яке значення від 1 до 6. Таким чином з однієї дисципліни для одного виду занять повинно стояти у розкладі k занять, де k розраховується за формулою:

$$k = (N + 1)/2 \qquad (1.1)$$

У формулі (1.1) передбачена операція ділення без остачі. Якщо N – парне число, то всі заняття будуть проводитися щотижня; інакше одне заняття буде проводитися через тиждень.

Як правило, для проведення всіх k занять одного виду з однієї дисципліни в одній академічній групі призначається один викладач. Але у зв’язку з об’єднанням у навчальних планах деяких навчальних дисциплін в одну дисципліну можлива ситуація, коли цю об’єднану дисципліну ведуть декілька провідних викладачів послідовно або паралельно. Випадок ведення паралельних занять з однієї дисципліни декількома викладачами простіше урахувати в програмі автоматизованого складання розкладу занять, якщо при цьому не збільшується кількість занять, що проводяться через тиждень. Оскільки розрахунковою одиницею при складанні розкладу занять є одна навчальна пара, до якої прикріплюється викладач, то у цьому випадку можна до кожного з k занять прикріпити окремого викладача без змін у алгоритмі розрахунку розкладу занять.

Випадок послідовного проведення заняття різними викладачами протягом семестру є більш складним, і в даній роботі він не розглядається.

2 МЕТОДОЛОГІЯ ФУНКЦІОНАЛЬНОГО МОДЕЛЮВАННЯ

2.1 Розробка контекстної діаграми IDEF0

IDEF0 – Function Modeling – методологія функціонального моделювання і графічного описання процесів, призначена для формалізації і опису бізнес-процесів. Особливістю IDEF0 є її акцент на ієрархічне представлення об'єктів, що значно полегшує розуміння предметної області. В IDEF0 розглядаються логічні зв'язки між роботами, а не послідовність їх виконання в часі (WorkFlow) [1 – 4].

Контекстна діаграма має один блок, що описує функцію одного рівня, управління, входи та виходи, а також мету моделі та точку зору, з якою будується модель.

Забезпечення навчального процесу ЗВО є головною функцією інформаційної системи, що моделюється. Розробка розкладу занять є складовою частиною навчального процесу.

Для складання розкладу занять використовуються нормативні документи МОН та ЗВО, які містять вимоги до навчальних планів зі спеціальностей, навчальні плани факультетів, контингент студентів на поточний семестр, дані про закріплення навчальних планів за академічними групами, розподіл навчальних дисциплін між кафедрами, розподіл занять між викладачами кафедри, дані про аудиторний фонд ЗВО.

Список вхідних даних: аудиторний фонд, нормативні документи МОН та ЗВО, дані студента, дані викладача, навчальний план, контингент студентів.

Але всі ці перелічені вихідні дані стосуються процесу складання розкладу занять, відокремленого від загального навчального процесу університету. Якщо розглядати розробку розкладу занять як єдиний процес – від розробки навчальних планів до отримання розкладу занять, – то вихідними даними для цього єдиного процесу будуть дані про студентів та викладачів. Нормативні документи та дані про аудиторний фонд відносяться до управління процесом.

Список функцій: розробка навчальних планів за спеціальностями, розподіл занять між викладачами, складання розкладу занять.

Контекстна діаграма IDEF0 розробки розкладу занять зображена на рис.2.1.

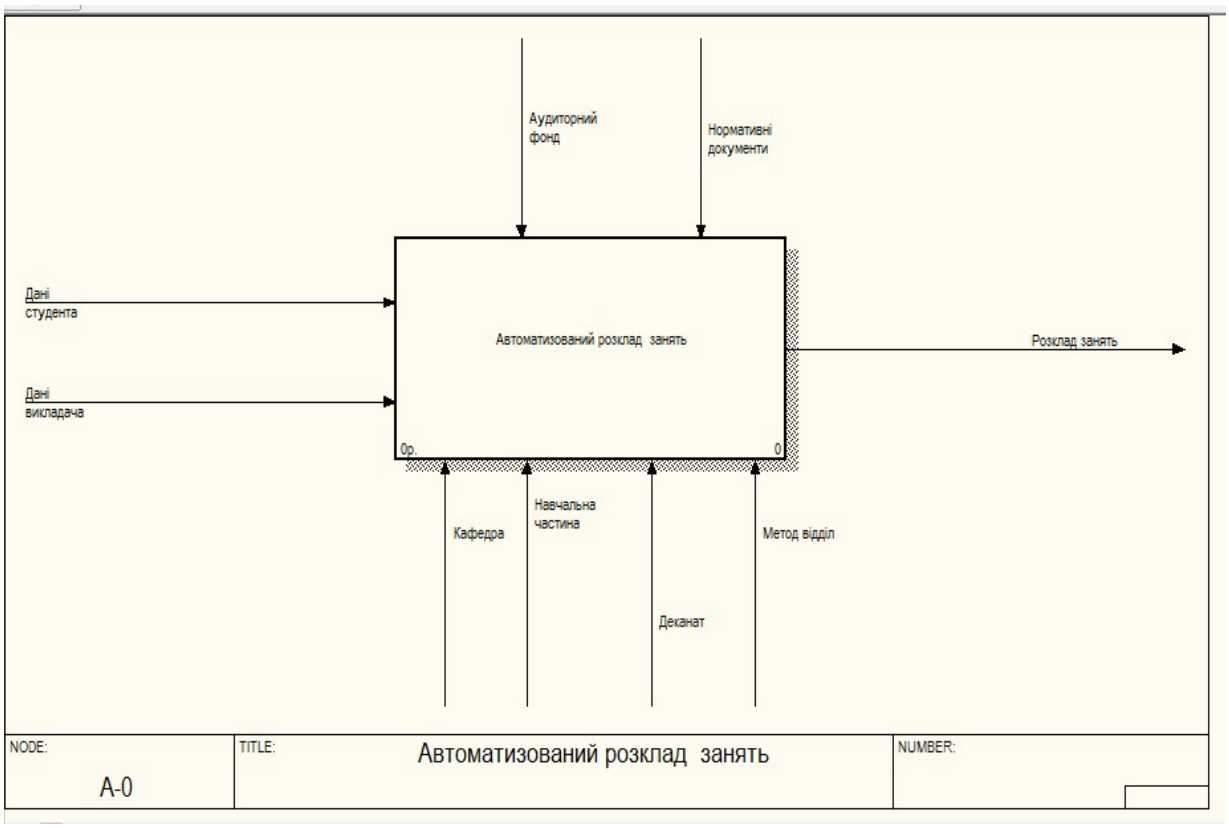

Рисунок 2.1 – Контекстна діаграма IDEF0

Стрілки контекстної діаграми:

1) Входи: дані студента, дані викладача.
2) Управління: аудиторний фонд, нормативні документи.
3) Механізми: деканат, методичний відділ, кафедра, навчальна частина.
4) Виходи: розклад занять.

2.1 Декомпозиція контекстної діаграми IDEF0

Згідно стандарту IDEF0 на кожному рівні декомпозиції треба використовувати принцип обмеження об'єкта, тому вважається, що єдиний блок і декілька стрілок на контекстному рівні використовуються для визначення кордону всієї системи. Відповідно, стрілки, які відносяться до цього блоку, описують головні управління, входи, виходи і механізми цієї системи.

Декомпозиція контекстної діаграми IDEF0 повинна включати блоки розробки навчальних планів, розподілу навчального навантаження між викладачами, складання розкладу занять.

На рис. 2.2 зображено роботу системи «Автоматичний розклад занять».

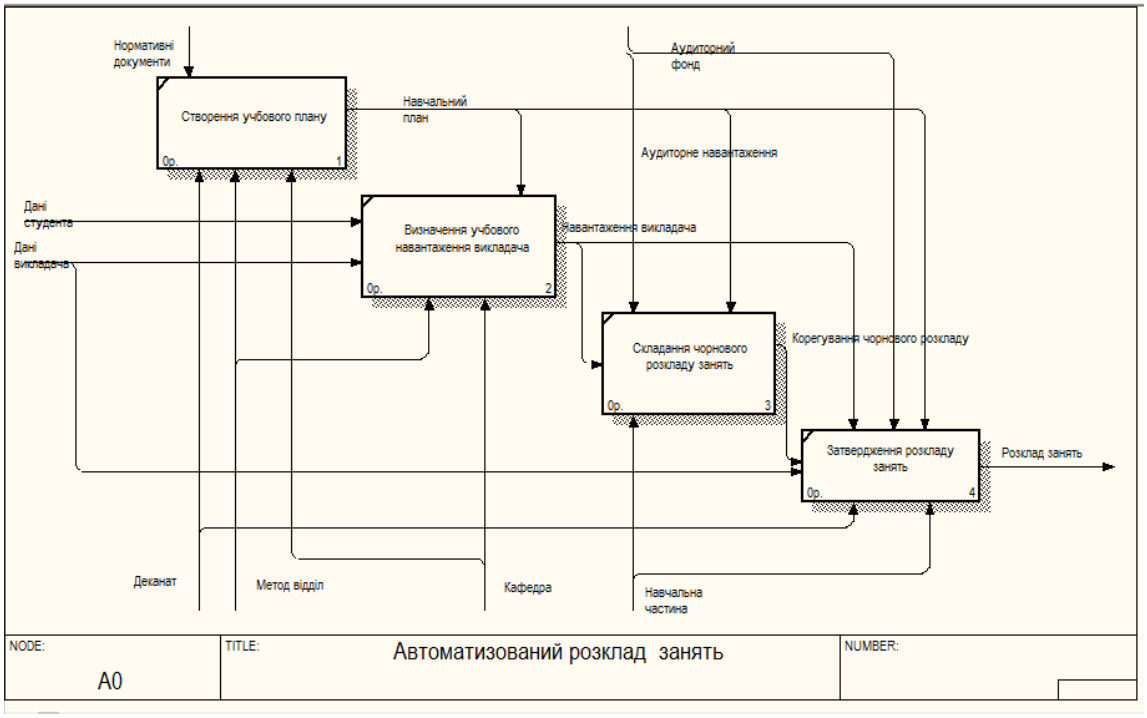

Рисунок 2.2 – Діаграма декомпозиції IDEF0

Методологія послідовного виконання процесу використовується для деталізації процесів, що відбуваються у середині блоків діаграми декомпозиції IDEF0.

Стандарт IDEF3 (англ. Integrated DEFinition for Process Description Capture Method) – методологія моделювання і стандарт документування процесів, що відбуваються в системі. Метод документування технологічних процесів є механізм документування та збору інформації про процеси. IDEF3 показує причинно-наслідкові зв'язки між ситуаціями і подіями в зрозумілій експерту формі, використовуючи структурний метод вираження знань про те, як функціонує система, процес або підприємство.

IDEF3 широко застосовується при розробці інформаційних систем. При цьому використовується інструмент візуального моделювання бізнес-процесів. Знання про процеси структуровані у вигляді контекстних сценаріїв, що робить IDEF3 зручним інструментом збору даних для опису системи.

3 МЕТОДОЛОГІЯ МОДЕЛЮВАННЯ ПОТОКІВ ДАНИХ

Діаграма потоків даних DFD (англ. Data Flow Diagram) – модель проектування, графічне представлення «потоків» даних в інформаційній системі. Діаграма потоків даних також може використовуватись для візуалізації процесів обробки даних (структурне проектування).

Вважається звичним спершу креслити контекстну діаграму потоків даних, завдяки чому буде показано взаємодію системи із зовнішніми модулями. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати розроблювану систему.

Діаграми потоків даних містять чотири типи графічних елементів:

1) процеси – являють собою трансформацію даних в рамках описуваної системи;
2) сховища даних (репозиторії);
3) зовнішні по відношенню до системи сутності;
4) потоки даних між елементами трьох попередніх типів.

Контекстна діаграма потоків даних містить нульові процеси з іменами, що відображають діяльність організації, зовнішні сутності, з'єднані з нульовим процесом за допомогою потоків даних. Потоки даних відповідають документам, запитам або повідомленнями, якими зовнішні сутності обмінюються з організацією (рис. 3.1).

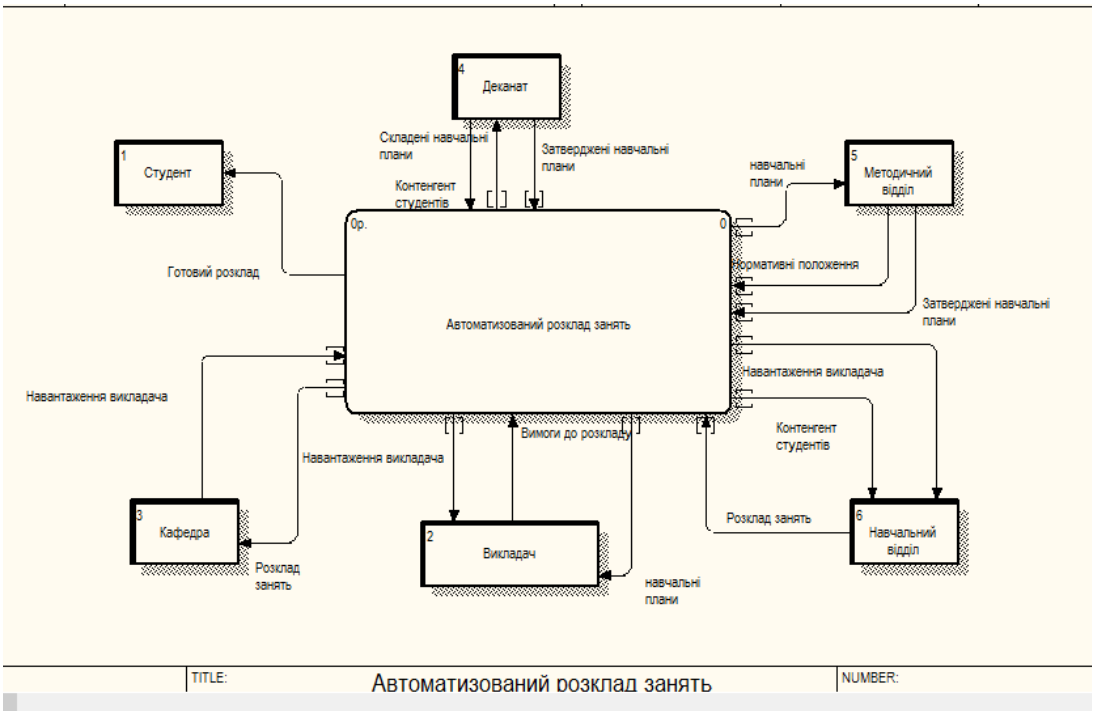

Рисунок 3.1 – Контекстна діаграма DFD автоматизованого розкладу занять

Наступним кроком моделювання потоків даних є декомпозиція контекстної діаграми DFD. Нульовий процес розбивається на складові системи і підпроцеси, які звичайно відповідають блокам декомпозиції діаграми IDEF0. Уточнюються потоки даних, які існують між зовнішніми сутностями та процесами. На рис. 3.2. зображена діаграма декомпозиції DFD автоматизованого розкладу занять.

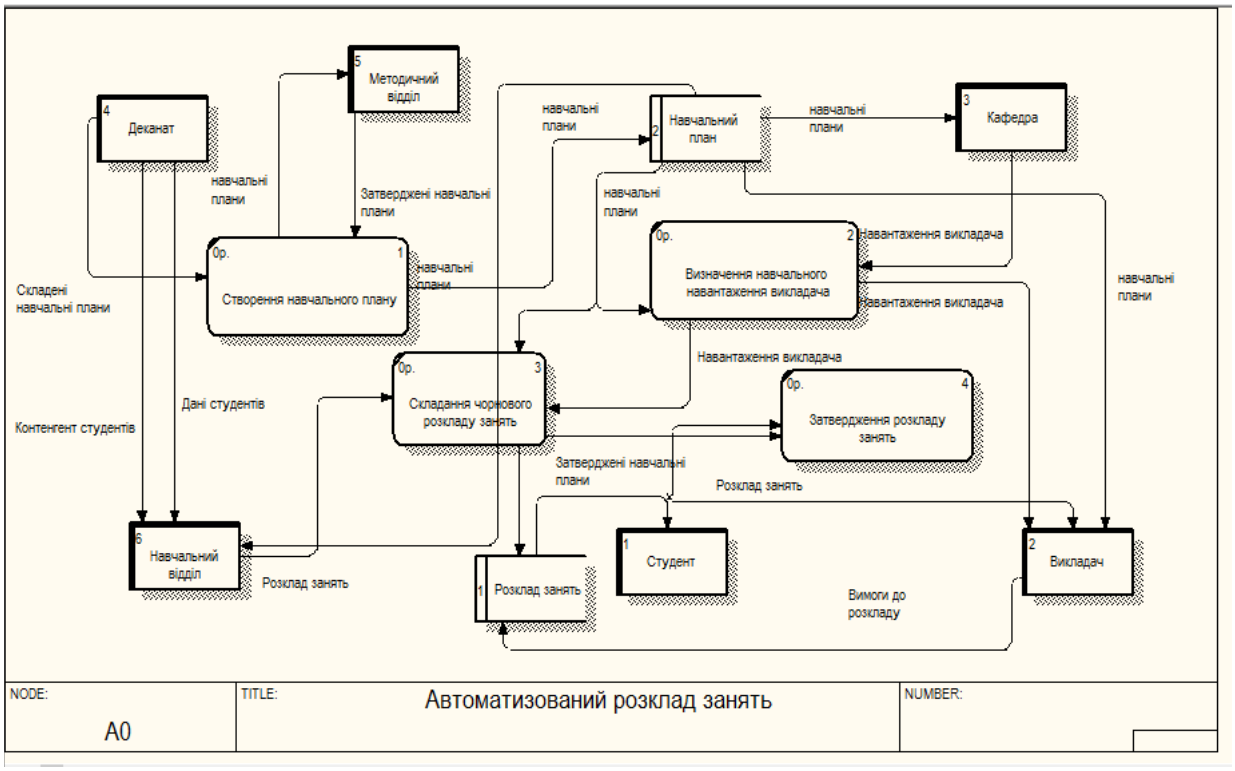

Рисунок 3.2 – Діаграма декомпозиції DFD автоматизованого розкладу занять

4 ФОРМУЛЮВАННЯ ЗАДАЧІ СКЛАДАННЯ РОЗКЛАДУ ЗАНЯТЬ В ТЕРМІНАХ ТЕОРІЇ ВІДНОШЕНЬ І РЕЛЯЦІЙНОЇ АЛГЕБРИ

4.1 Загальний випадок занять розміром дві академічні години на тиждень

У термінах теорії відношень та реляційної алгебри (алгебри відношень) [5 – 7] добре формулюються як опис структури бази даних «Розклад занять», так і процес автоматизованого складання розкладу занять.

Назвемо «заняттям» одну навчальну пару в одній академічній групі по одному предмету, що проводиться у одній аудиторії. Опустимо в цій постановці задачі такий вид занять, як лабораторні заняття з розділом групи на підгрупи, та потокові лекції, які проводиться в одній і тій же аудиторії одним викладачем для деякого набору (потоку) груп.

Нехай відношення G містить список усіх академічних груп у ЗВО, відношення D містить список усіх предметів (навчальних дисциплін), що читаються в цьому ЗВО, а відношення H містить список різних видів занять (лекції, практичні заняття, лабораторні роботи). Тоді операцією декартового (прямого) добутку отримаємо відношення W, яке міститиме список усіх можливих занять усіх груп:

$$W = G \times D \times H \tag{4.1}$$

Проте в університеті студенти не вивчають усі можливі навчальні дисципліни, крім того не по всім навчальним дисциплінам передбачені всі види занять. Кожен факультет має набір навчальних планів, і кожна академічна група вчиться за деяким навчальним планом.

Останні роки у студентів з’явилась можливість самим формувати собі набір дисциплін, які відносяться до вибіркових навчальних дисциплін. Таким чином, вивчення однакових навчальних дисциплін всією академічною групою є правилом лише для обов’язкових дисциплін навчального плану, який закріплений за групою. Але урахування індивідуальних навчальних планів студентів у процесі складання розкладу занять не відноситься до завдання даної магістерської роботи, тому будемо вважати одиницею, яка отримує знання, не студента, а академічну групу. Це припущення має право на існування, тому що більшість дисциплін навчальних планів відноситься до обов’язкових і вивчаються всіма студентами академічної групи. Урахування

індивідуальних навчальних планів студентів у процесі складання розкладу занять потребує більш складної структури бази даних з додаванням асоціативних типів сутностей, які відстежують відмінності індивідуального плану кожного студента від загального навчального плану академічної групи. Ця задача потребує додаткових досліджень.

Отже, будемо вважати, що всі студенти однієї академічної групи здобувають вищу освіту за одним навчальним планом зі спеціальності. Навчальний план має декілька наборів вибіркових дисциплін, які визначають спеціалізації всередині спеціальності. Кожна академічна група прикріплюється не лише до навчального плану зі спеціальності, але також до набору дисциплін зі спеціалізації.

Опишемо у базі даних у якості типу сутності НАВЧАЛЬНИЙ_ПЛАН саме сукупний навчальний план зі спеціальності та спеціалізації. Тоді можна вважати, що кожна академічна група навчається за деяким конкретним навчальним планом – екземпляром сутності вказаного типу сутності.

Таким чином, у формулі (4.1) замість прямого добутку відношень G, D, H мають бути операції природного з'єднання цих відношень з відношенням F, що є відображенням типу сутності Навчальний_план:

$$W = G \bowtie ((F \bowtie D) \bowtie H) \qquad (4.2)$$

Крім того, в університеті існує множина викладачів, список яких може бути представлений відношенням T. Тоді множина занять із закріпленими за ними викладачами, міститься у відношенні S, яке обчислюється за формулою:

$$S = W \bowtie T, \qquad (4.3)$$

Між відношеннями T і W використовується операція природного з'єднання, оскільки передбачається, що між типами сутностей Викладач і Заняття існує тип зв'язку «один до багатьох» (а не «багато до багатьох»), тобто кожне заняття приписується до одного конкретного викладача.

Нехай відношення A містить список усіх аудиторій закладу вищої освіти, в яких можуть проводитись заняття, відношення P містить список усіх навчальних пар протягом тижня, на яких можуть проводитись заняття в цьому ЗВО.

Тоді список усіх можливих варіантів розкладу занять буде описуватись відношенням Z, яке знаходиться за допомогою операції декартового добутку над відношеннями S, A та P:

$$Z = S \times A \times P \tag{4.4}$$

Серед занять ЗВО існує велика кількість занять, які можуть проводитись лише у спеціалізованих аудиторіях, наприклад, комп'ютерних класах, або різних лабораторіях. Крім того, аудиторії відрізняються за кількістю посадкових місць, тому для проведення заняття потрібно вибирати ті аудиторії, в яких достатня кількість місць для даної академічної групи. Таким чином необхідне похідне відношення (асоціативний тип сутності) SA (Study Auditory – аудиторії для заняття), що визначає множину аудиторій, в яких може проводитись кожне заняття:

$$Z = ((S \bowtie SA) \bowtie A) \times P \tag{4.5}$$

Крім того, не всі заняття можуть проводитись на будь-якій навчальній парі тижня. Для академічних груп можуть призначатись дні для самостійної підготовки студентів, крім того, можуть призначатись дні тижня для спеціалізованих робіт або занять, які не описуються загальним розкладом занять у вищі, наприклад, чергування у яких-небудь спеціалізованих лабораторіях.

Деякі викладачі зайняті у деякі дні тижня, або на деяких навчальних парах виконанням інших робіт, не пов'язаних з проведенням занять. Наприклад, викладача повинен бути присутнім на засіданнях вченої ради університету, або виконує інші громадські обов'язки. Тому зі списку допустимих для проведення занять навчальних пар викладачів повинні виключатись навчальні пари, на яких він виконує інші обов'язки.

Також існують деякі заняття, які не відповідають наведеному вище визначенню заняття. Наприклад, у ЗВО вивчаються декілька іноземних мов. Групи для вивчення студентами іноземних мов можуть складатись зі студентів декількох академічних груп, при цьому студенти однієї академічної групи можуть опинитись у різних групах для вивчення іноземної мови. У таких випадках потрібно виділити навчальну пару, на якій будуть проводитись заняття з іноземних мов для деякого академічного потоку. Але для цього потоку не призначається для кожного академічної група (або її

підгрупи) конкретний викладач та конкретна аудиторія, а призначається множина викладачів, які будуть проводить заняття для деякої підмножини студентів потоку, та множина аудиторій, в яких ці заняття будуть проводитись. Розподіл студентів та аудиторій між викладачами виконується відповідальним працівником кафедри іноземних мов (або кафедри, за якою закріплені заняття з іноземних мов).

Зазвичай такі заняття визначаються навчальним відділом до початку розрахунку загального розкладу занять. Таким чином, деякі академічні групи, викладачі та аудиторії вже будуть зайняті на деяких навчальних парах перед початком розрахунку розкладу занять.

Всі перераховані чинники вимагають наявності у базі даних асоціативних типів сутностей, які зберігають інформацію про допустимі для постановки занять у розклад навчальні пари викладачів, академічних груп та аудиторій.

Заняття можна поставити на навчальну пару, яка одночасно вільна для групи, викладача та аудиторії. Така навчальна пара буде допустимою для проведення даного заняття. Тому у формулі (4.5) замість операції декартового добутку з відношенням P повинна стояти операція природного з'єднання заняття зі списком навчальних пар, які допустимі для його проведення:

$$Z = (((W \bowtie SA) \bowtie A) \bowtie PA) \bowtie P \bowtie (PG \bowtie PT), \quad (4.6)$$

де PA – похідне відношення, яке визначає допустимі навчальні пари для аудиторій;

PG – похідне відношення, яке визначає допустимі навчальні пари для академічних груп;

PT – похідне відношення, яке визначає допустимі навчальні пари для викладачів.

Рішенням задачі складання розкладу занять є вибірка з відношення Z відношення V – реального розкладу. У відношення V кожне заняття входить тільки один раз, тобто призначене для проведення на якій-небудь одній навчальній парі в якій-небудь одній конкретній аудиторії.

При цьому для відношення V повинні виконуватися наступні вимоги:

– кожне заняття з відношення S повинне міститися у відношенні V і тільки один раз;

- кожен викладач повинен мати рівно стільки годин занять в тиждень, скільки передбачено для нього за планом;
- не може бути призначено більше за одне заняття для будь-якої групи на одну навчальну пару;
- не може бути призначено більше за одне заняття для будь-якого викладача на одну навчальну пару;
- не може бути призначено більше за одне заняття для будь-якої аудиторії на одну навчальну пару.

З відношення Z можна вибрати багато варіантів реального розкладу занять, тобто відношення V знаходиться не однозначно. Різні варіанти розкладу занять відрізняються між собою ступенем комфортності розкладу для різних груп та викладачів. Не можна однозначно визначити оптимальний розклад занять, тому що не існує для нього чіткого визначення. Але зазвичай оптимальним вважається розклад занять, який задовольняє обов’язковим вимогам, що висуваються до розкладу занять у вищі, та який є досить комфортним для більшості академічних груп та викладачів.

Поняття комфортного розкладу занять є досить невизначеним і змінюється у залежності від багатьох факторів. Головним фактором є кількість наявних пар занять протягом тижня, які повинні бути у розкладі академічної групи або викладача. Якщо ця кількість велика – більше 15 пар на тиждень, – комфортним може вважатись розклад занять з 6 робочими днями на тиждень. Коли кількість навчальних пар у розкладі варіюється між 13 та 15 – комфортним може вважатись розклад з 4 робочими днями на тиждень. Якщо навчальних пар менше 13, за комфортний розклад слід вважати розклад з трьома робочими днями на тиждень.

Вільні дні тижня відводяться у викладачів під виконання інших видів діяльності: методична, наукова, виховна робота та виконання громадських обов’язків. У студентів вільні дні тижня відводяться під самостійну підготовку.

4.2 Урахування занять розміром одну академічну годину на тиждень

Найпростішим способом обліку такого виду занять є перетворення для розрахунку розкладу занять двох навчальних тижнів в один з наскрізною нумерацією навчальних пар протягом об'єднаного тижня. У цьому випадку кожне заняття, що проводиться щотижня, перетворюється у два заняття; заняття, що проводиться раз на тиждень, стає одним звичайним заняттям.

Головний недолік такого способу складання розкладу занять полягає в тому, що розклади занять на парному і непарному тижні семестру можуть не мати нічого спільного. На парному тижні можуть проводитися заняття з одних дисциплін, на непарному – з інших. За таким розкладу дуже важко вчитися студентам і проводити заняття викладачам.

Розклад занять потрібно складати для одного навчального тижня, але в ньому будуть присутні заняття з позначкою, що вони проводяться через тиждень. Для опису подібних позначок, як правило, використовують булевий (логічний) тип даних. Позначка може означати, що заняття проводиться щотижня. Тоді для звичайних занять ця позначка приймає значення "істина" (true), а для занять через тиждень – значення "брехня" (false). Але два значення позначки достатньо, щоб відмітити заняття, що потрібно поставити у розклад. Для заняття, що ставиться у розклад, потрібно три значення позначки: щотижня, на парному тижні, на непарному тижні. Тобто, при складанні розкладу занять з урахуванням занять через тиждень для подібних занять потрібно вибирати не тільки навчальну пару та аудиторію, а також тиждень проведення.

У базах даних передбачено, що дані можуть мати невизначене (порожнє) значення, яке умовно позначується через NULL. Це значення можна вибрати для позначки, що заняття у розкладі проводиться щотижня. Тоді значення true може означати парний тиждень, а значення false – непарний тиждень. Подібна позначка повинна використовуватись для опису вільних навчальний пар груп, викладачів, аудиторій.

Алгоритм постановки занять у розклад повинен враховувати, що заняття через тиждень можна ставити у розклад на пару, яка вільна або щотижня, або кожен тиждень; щотижневі заняття можна ставити тільки на пари, які вільні щотижня.

5 АЛГОРИТМ СКЛАДАННЯ РОЗКЛАДУ ЗАНЯТЬ

5.1 Огляд методів вирішення задачі складання розкладу занять

Пошуком розв’язку задачі автоматизованого складання розкладу занять останніми роками займалися багато авторів, якщо судити по великій кількості готових програмних продуктів цього виду. Проте, хоча в мережі Інтернет пропонується багато готових програм складання розкладу занять у закладах вищої освіти, але усі ці програми пропонуються до продажу із закритим кодом. Тому практично неможливо проаналізувати алгоритми, які в них використовуються. Відкриті публікації останніх років містять в основному рекомендації, які критерії необхідно враховувати в програмі складання розкладу занять для отримання оптимального варіанту розкладу занять, але не містять конкретного алгоритму обліку цих критеріїв [8 – 14].

При складанні алгоритмів вирішення задачі складання розкладу занять у більшості відкритих алгоритмів одним з основним параметрів алгоритму є ваговий коефіцієнт заняття, по якому проводиться сортування занять для визначення черговості постановки заняття в розклад.

Для обчислення цих вагових коефіцієнтів пропонуються різні формули, наприклад, з урахуванням оцінки свободи розташування заняття в розкладі [8,9]. Оцінка свободи розташування заняття в розкладі обчислюється різними авторами різними способами. В основному враховуються щільність занять в групі, щільність занять у викладача, кількість аудиторій, допустимих для проведення заняття.

Деякі автори пропонують враховувати інші критерії при обчисленні коефіцієнта свободи розташування заняття в розкладі [10 – 12]. Наприклад, кількість обмежень, які накладаються на розклад для різних викладачів: кількість навчальних днів на тиждень, конкретні дні проведення занять та інше.

Після обчислення у той або інший спосіб деякого коефіцієнта свободи розташування заняття в розкладі список занять упорядковується за збільшенням цього коефіцієнта.

Після проведення сортування списку занять в розклад в першу чергу додаються заняття, що знаходяться в голові списку, тобто з найменшою свободою розташування в розкладі. При додаванні занять в розклад здійснюється пошук найбільш вигідної, аудиторії і часу для проведення заняття. Для цього потрібний повний перебір варіантів проведення заняття у

просторі (аудиторія) та часі (номер пари та день тижня). В процесі перебору варіантів розташування заняття в першу чергу відбувається перевірка можливості проведення заняття за трьома умовами:

1) не відбувається «перекриття» занять. У випадку якщо воно сталося, заняття не може бути проведене;
2) аудиторія обладнана усім необхідним для проведення заняття, наприклад комп'ютерами, стендами для проведення експериментів, проектором і т. д.;
3) кількість робочих місць в аудиторії не менше кількості студентів в групі.

Потрібно відмітити, що перевірка перерахованих умов під час розрахунку розкладу занять є недоцільним. Друга та третя умови автоматично враховуються використанням похідного відношення SA (формули 4.5 – 4.6). Якщо в формулі (4.6) вважати, що похідні відношення PG, PT, PA містять дані про вільні пари груп, викладачів та аудиторій у поточний момент розрахунку, то перша умова також автоматично виконується.

У випадку, якщо обов'язкові умови виконуються, автори деяких алгоритмів пропонують оцінювати для можливих варіантів постановки заняття у розклад якість розташування заняття за наступними критеріями:

1) поява вікна в розкладі групи студентів;
2) поява вікна в розкладі викладача;
3) надмірність кількості місць в аудиторії по відношенню до кількості студентів;
4) проведення заняття в невдалий час, наприклад четвертим або п'ятим по рахунку цього дня для цієї групи студентів;
5) зникнення вікна в розкладі групи студентів;
6) зникнення вікна в розкладі викладача;
7) зникнення вікна в розкладі використання аудиторії.

Яким чином з набору перерахованих критеріїв виконується вибір варіанту з самим високим показником якості розташування заняття, автори не уточнюють.

Деякі автори вказують, що критерії оцінки якості розташування заняття в розкладі використовуються при обчисленні цільової функції. Формули, по яких обчислюється цільова функція, при цьому не наводяться.

Загалом алгоритм складання розкладу занять можна представити таким чином:

1) для першого заняття в відсортованому списку обчислюються коефіцієнти якості розташування в розкладі для усіх можливих пар занять і аудиторій;
2) для подальших занять в відсортованому списку також обчислюються коефіцієнти якості розташування в розкладі для усіх можливих пар занять і аудиторій з урахуванням знаходження в розкладі занять, що розташовані вище в списку;
3) по коефіцієнтах якості розташування занять в розкладі обчислюється цільова функція;
4) для знаходження оптимального розкладу визначається те розташування занять в розкладі, яке мінімізує цільову функцію.

Авторами запропонованих алгоритмів не уточнюється, чи відбувається обчислення цільової функції і її мінімізація тільки один раз, після перебору усіх можливих станів розкладу занять, чи існує декілька ітерацій складання розкладу, на кожній з яких оптимізується розклад за умови додавання декількох занять зі списку до занять, що вже закріплені в розкладі на попередній ітерації.

Обидва вказані підходи мають свої недоліки. У першому випадку виходить багатовимірний масив критеріїв якості розташування занять, при цьому для кожного заняття існує множина станів в тривимірному просторі (навчальна пара, аудиторія, критерій якості). Знаходження рішення подібної задачі видається проблематичним. В публікаціях останніх років автори пропонують додавати нові критерії визначення якості розташування заняття у розкладі: з урахуванням переходу в продовж одного дня академічних груп з однієї аудиторії в іншу та з одного навчального корпусу до іншого; з урахуванням взаємного розташування занять, що належать до гуманітарних наук та до точних наук; з урахуванням взаємного розташування лекцій та практичних занять і т.п. Збільшення критеріїв визначення якості розташування заняття у розкладі приводить до підвищення складності вирішення задачі.

У другому випадку рішення задачі знайти простіше, але не можна гарантувати, що воно буде оптимальним. Більше того, в цьому випадку може виникнути тупикова ситуація, коли усі можливі для деякого заняття положення у двовимірному просторі (навчальна пара, аудиторія) вже зайняті іншими заняттями на попередніх ітераціях складання розкладу.

5.2 Вибір ітераційного алгоритму автоматичного складання розкладу занять

Як вказувалось вище, задача складання оптимального розкладу занять практично не має рішення, оскільки не можливо сформулювати поняття оптимального розкладу занять. В це поняття різні автори вкладають різний сенс, критерії оптимальності частіше описуються як деякі якісні характеристики, а не у вигляді кількісних атрибутів. Тому неможливо чітко сформулювати вимоги до цільової функції, мінімізація якої дозволить отримати оптимальний розклад занять. Навіть у випадках, коли автори наводять вигляд цільової функції, не зрозуміло, чи буде екстремум цієї цільової функції відповідати оптимальному розкладу занять. Якщо вважати, що оптимальним є той розклад занять, для якого задовольняється максимальна кількість вимог викладачів до розкладу, то як знаходити максимум для великої кількості викладачів, вимоги яких можуть суперечити один одному.

Більш простим рішенням для поставленої задачі є використання ітераційного алгоритму, якій на кожному кроці ітерації знаходить найбільш зручне розташування для чергового набору занять.

Перед кожним кроком ітерації виконується ранжирування списку занять за збільшенням коефіцієнта свободи розташування заняття в розкладі. Цей коефіцієнт обчислюється на кожному кроці як деяка змінна, яка розраховується з урахуванням кількості навчальних пар, на які можна ще поставити кожне заняття.

Як правило, передбачається, що при задаванні досить слабких обмежень на розклад в сенсі його зручності для студентів і викладачів, а також наявності достатнього аудиторного фонду університету, задача складання розкладу занять має рішення. Але це рішення дуже далеке від досконалості, тому передбачається, що воно потім допрацьовується вручну користувачем-диспетчером.

Для отримання більш зручного розкладу занять потрібно в програмі враховувати велику кількість обмежень у вигляді вихідних даних для складання зручного розкладу для студентів і викладачів. Зазвичай у такій постановці задача не має вирішення, оскільки виникає тупикова ситуація: для якихось занять немає жодної навчальної пари, коли одночасно були б вільні і група, і викладач, і хоча б одна допустима для заняття аудиторія.

Після виникнення тупикової ситуації програма зупиняє свою роботу і чекає від користувача зміни вихідних даних у бік послаблення поставлених обмежень. Після чого програма запускається знову для продовження розрахунків.

У разі задавання слабких обмежень на розклад звичайний розподіл занять по аудиторіях відбувається більш менш рівномірно, навіть за наявності аудиторій, яким викладачі віддають перевагу. Таким чином, аудиторії, що найбільш зажадалися викладачами, можуть виявитися недостатньо завантаженими, але при цьому заняття проходитимуть в аудиторіях, що менш пристосовані для проведення заняття з точки зору викладача.

У разі задавання досить жорстких обмежень допускається вказівка для кожного заняття дуже обмеженої множини допустимих аудиторій. В цьому випадку найбільш обладнані і зручні аудиторії будуть затребувані в першу чергу, і з великою вірогідністю може виникнути тупикова ситуація при вирішенні задачі складання розкладу занять.

Для запобігання тупикової ситуації при автоматизованому складанні розкладу занять за умови дотримання жорстких обмежень можна на кожному кроці ітерації ставити у розклад лише одне заняття, що має мінімальну кількість варіантів постановки у розклад. Для більш визначеного знаходження такого заняття потрібно знайти алгоритм сортування списку занять для постановки у розклад.

Використання якогось одного складного емпіричного коефіцієнту, за яким буде проводитись сортування списку занять, не є доцільним, оскільки можна використовувати багаторівневе сортування.

5.3 Визначення змінних коефіцієнтів сортування занять у списку

На початку розрахунку розкладу занять у похідні відношення PT, PG, PA записується інформація про допустимі для постановки занять у розклад навчальні пари груп, викладачів та аудиторій відповідно. При постановці кожного заняття у відповідних групи, викладача та аудиторії зменшується кількість вільних пар. Наявні вільні пари можна знаходити операцією вибірки з використанням відношень PT, PG, PA та відношення STT (Studi in TimeTable), яке містить дані про поставлені у розклад заняття.

Для зменшення кількості вибірок з таблиць бази даних і зменшення часу роботи програми бажано мати в схемі бази даних відношення, що описують вільні пари для груп, викладачів, аудиторій.

Нехай FPT (Free Pare Teacher) означає відношення, що містить список вільних пар викладачів, FPG (Free Pare Group) – відношення, що описують вільні пари академічних груп, а FPA (Free Pare Audutory) – відношення, що описують вільні пари аудиторій. Відношення FPT, FPG, FPA матимуть тільки по два атрибути – ідентифікатор навчальної пари і ідентифікатор об’єкту (викладача, групи або аудиторії).

На початку розрахунку розкладу занять у відношення FPT, FPG, FPA записуються дані з відношень PT, PG, PA відповідно. Дані для відношення PT формуються з вимог до розкладу занять викладачів, що надають кафедри, та доповнюється при завданні навчальним відділом нестандартних видів занять, наприклад, занять з іноземних мов (розділ 4). Дані для відношень PG, PA формуються при введенні користувачем навчального відділу інформації про нестандартні види занять та призначені для академічних груп дні самостійної підготовки студентів.

На кожному кроці розрахунку розкладу занять з відношень FPT, FPG, FPA видаляються кортежі, які відповідають викладачеві, групі та аудиторії заняття, яке ставиться у розклад.

Заняття, які необхідно поставити в розклад, описуються відношенням Study. При постановці якогось заняття в розклад навчальна пара для нього може вибиратися тільки з множини навчальних пар, що є перетином множини вільних пар у групи, для якої проводиться заняття, у викладача, який проводить заняття, і для аудиторії, в якій проводитиметься заняття.

Таким чином, маємо набір відношень:

Study (id_st, id_g, id_sub, id_kind, id_t, is_tt)

FPG (id_g, id_p)

FPT (id_t, id_p)

FPA (id_a, id_p)

SA (id_st, id_a)

Атрибути відношень:

id_st – ідентифікатор заняття;

id_g – ідентифікатор групи;

id_sub – ідентифікатор дисципліни;

id_kind – ідентифікатор виду занять (лекції, практичні, лабораторні)

id_t – ідентифікатор викладача;

id_p – ідентифікатор навчальної пари;

id_a – ідентифікатор аудиторії;

is_tt – логічний атрибут, що описує, поставлено вже заняття в розклад (is_tt =1) або ні (is_tt =0).

Можливими парами для проведення заняття заданого викладача в заданій групі будуть навчальні пари, коли одночасно вільні і академічна група і викладач. Природне з'єднання відношень FPG і FPT містить навчальні пари, на яких кожен викладач може проводити заняття з кожною групою:

$$\text{Temp1}(\text{id\_t}, \text{id\_g}, \text{id\_p}) \leftarrow \text{FPG} \bowtie \text{FPT}, \qquad (5.1)$$

У формулі (5.1) з'єднання виконується за загальним атрибутом id_p. Тільки деякі кортежі з відношення Temp1 мають сенс в процесі створення розкладу занять, оскільки викладачі не ведуть заняття в усіх академічних групах ЗВО.

Список занять з допустимими для них аудиторій отримуємо природним з'єднанням відношень Study та SA (5.2):

$$\text{Temp2}(\text{id\_st},\text{id\_g},\text{id\_sub},\text{id\_kind},\text{id\_t},\text{is\_tt},\text{id\_a})) \leftarrow \text{Study} \bowtie \text{SA} \qquad (5.2)$$

У формулі (5.2) з'єднання виконується за загальним атрибутом id_st.

Список всіх занять, які потрібно поставити у розклад, з усіма варіантами їх розташування у двомірному просторі (пара, аудиторія) міститься у відношенні Temp3 (id_st,id_g,id_sub,id_kind,id_t,is_tt,id_a,id_p), яке отримуємо з формули (5.3):

$$\text{Temp3} \leftarrow (\text{Temp2} \bowtie \text{FPA}) \bowtie \text{Temp1} \qquad (5.3)$$

У формулі (5.3) з'єднання Temp2 з FPA виконується за загальним атрибутом id_a, а результат цього з'єднання з'єднується з відношенням Temp1 за атрибутами id_t, id_g, id_p.

Варіанти розташування конкретного заняття з ідентифікатором id_st = i у двомірному просторі (пара, аудиторія) отримуємо операцією вибірки (5.4):

$$\text{Temp4}(\text{id\_st},\text{id\_g},\text{id\_sub},\text{id\_kind},\text{id\_t}, \text{id\_a},\text{id\_p}), \leftarrow \sigma_{\text{id\_st} = i}(\text{Temp3}) \qquad (5.4)$$

Коефіцієнтом свободи (K2) розташування заняття з ідентифікатором id_st = i у двомірному просторі (пара, аудиторія) буде кількість кортежів у відношенні Temp4.

Для знаходження кількості навчальних пар, на які можна поставити задане заняття, тобто коефіцієнта свободи розташування заняття в одномірному просторі навчальних пар (K1), потрібно знайти відношення Temp5 з атрибутами id_st, id_g, id_sub, id_kind, id_t, id_p, за допомоги проекції відношення Temp4 на всі атрибути, крім id_a:

$$Temp5 \leftarrow \pi_{id\_st,id\_g,id\_sub,id\_kind,id\_t,\ id\_p}(Temp4) \qquad (5.5)$$

Коефіцієнт K1 для заданого заняття дорівнює кількості кортежів у відношенні Temp5. Якщо на деякій навчальній парі задане заняття можна поставити у розклад у N аудиторій, така навчальна пара буде записана у одному кортежі відношення Temp5 та у N кортежах відношення Temp4.

На перших кроках складання розкладу усі заняття матимуть великі значення коефіцієнтів K1 та K2, оскільки вільні усі аудиторії, усі групи, усі викладачі. Деяка відмінність у величині коефіцієнту K1 для різних занять обумовлюється завданням навчальних пар, на які не можуть ставитися заняття для заданих груп або викладачів. Величина коефіцієнту K2 залежить ще і від кількості допустимих для проведення заняття аудиторій.

Проте, на початку складання розкладу бажано враховувати і інші критерії першочерговості постановки заняття в розклад, окрім коефіцієнтів K1 та K2, наприклад, щільність занять у викладачів.

Тому коефіцієнти K1 та K2 мають бути визначальними при ранжируванні занять в списку тільки при зменшенні їх до деяких заданих величин K1MAX та K2MAX, наприклад, коли вільних пар для заняття залишається не більше 5, а кількості місць розташування заняття у двомірному просторі (пара, аудиторія) залишається менше 10. Тобто, у розрахунках використовуються коефіцієнти K1_0 та K2_0:

K1_0 = min (K1, K1MAX)

K2_0 = min (K2, K2MAX)

Таким чином великих значеннях коефіцієнтів K1 та K2 вони становляться константами, і при сортуванні занять в списку в першу чергу враховуються інші критерії.

Константи K1MAX та K2MAX зберігаються в БД у таблиці констант. Ці константи можна змінювати перед початком розрахунку розкладу занять.

5.4 Критерії сортування занять в списку для складання розкладу

Які ще критерії бажано враховувати в алгоритмі визначення черговості постановки занять в розклад?

Важливо враховувати завантаженість викладача і можливість урахування його вимог і обмежень до розкладу. При цьому краще враховувати не абсолютну завантаженість викладача, тобто кількість пар його занять на тиждень за планом, а відносну завантаженість, тобто відношення кількості ще не розподілених пар його занять до кількості вільних навчальних пар, на які можна поставити заняття цього викладача.

Кількість занять викладача (KTst) з ідентифікатором id_t = j, які ще потрібно поставити у розклад, дорівнює кількості кортежів відношення Temp6, яке є вибіркою з відношення Study кортежів за вказаною умовою:

$$\text{Temp6 (id\_st, id\_g, id\_sub, id\_kind)} \leftarrow \sigma_{\text{id\_t} = \text{j}}(\text{Study}) \qquad (5.6)$$

Кількість вільних навчальних пар цього викладача (KTp), на які можна поставити вказані заняття, дорівнює кількості кортежів відношення Temp7, яке є вибіркою з відношення Temp5 кортежів за вказаною умовою:

$$\text{Temp7 (id\_st, id\_g, id\_sub, id\_kind, id\_p)} \leftarrow \sigma_{\text{id\_t} = \text{j}}(\text{Temp5}) \qquad (5.7)$$

Для визначення свободи розташування заняття у розкладі викладача можна використовувати дві величини: відносну свободу розташування заняття K3 = (KTp – KTst), та щільність решти занять у розкладі викладача K4, де K4 = KTst/KTp.

В першу чергу слід враховувати K3, потім – K4. Для урахування щільності занять для викладачів з різним (але досить великим) K3 можна встановити межу врахування K3 і застосовувати коефіцієнт K3_0:

K3_0 = min (K3, K3MAX)

Константа K3MAX також повинна зберігатись у таблиці констант БД.

Розглянуті у розділі 5.1 алгоритми автоматизованого складання розкладу занять з усіх умов, що враховуються в розрахунках, формували деякий один коефіцієнт, по якому і проводилося сортування занять в списку для складання розкладу. Проте в сучасних системах управління базами даних можуть використовуватися багаторівневі сортування, тому немає необхідності вигадувати формулу, в якій з декількох величин шляхом деяких

алгебраїчних операцій розраховується єдиний коефіцієнт сортування занять. Можна встановити черговість обліку різних критеріїв ранжирування занять, і включити ці критерії у багаторівневе сортування списку занять.

5.5 Загальний алгоритм складання розкладу занять

Розклад занять можна розраховувати у декілька етапів. На кожному етапі у розклад ставиться задана множина занять, наприклад, заняття академічних груп заданого факультету, або академічних груп заданого курсу всіх факультетів, або академічних груп заданого курсу заданого факультету. Після закінчення розрахунку розкладу для вибраної множини занять можливе редагування отриманого розкладу за допомогою програми виду «АРМ диспетчеру навчального відділу». Корегувати варто лише розклад занять академічних груп, а не розклад викладачів, оскільки останній може ще змінюватись коли в нього будуть додаватись заняття академічних груп з наступної вибраної множини занять для розрахунку розкладу.

Потім вибирається наступна множина занять для розрахунку розкладу. Процес повторюється, доки всі заняття всіх академічних груп не будуть поставлені у розклад.

Вибір поточної множини занять для постановки у розклад повинен виконувати користувач. Для цього у клієнтському додатку потрібно передбачити відповідний інтерфейс.

Програмно цей вибір виконується дуже просто – замість відношення Study в процесі розрахунку розкладу занять використовується відношення ST, яке є вибіркою з відношення Study і розраховується за формулою (5.8):

$$ST \leftarrow \sigma_{id\_f = k \wedge curs = m}(Study \triangleright Groups) \qquad (5.8)$$

Відношення Groups містить список всіх академічних групи ЗВО. У формулі (5.8) при виборі множини занять для постановки у розклад використовуються атрибути id_f та curs відношення Groups: атрибут id_f є ідентифікатором факультету, до якого належить група; атрибут curs є номером курсу, на якому навчається група. Таким чином, у формулі (5.8) для постановки у розклад вибираються заняття всіх академічних груп, які належать до факультету з ідентифікатором k та навчаються на курсі з номером m.

Для всіх занять з відношення ST розраховуються коефіцієнти K1, K2, K3, K4. Визначається, чи є у відношенні ST заняття викладачів з дуже щільним розкладом, тобто заняття, для яких K3<K3_0. Якщо такі заняття є, і немає занять з K1 = 1, то список занять упорядковується наступним чином:

1) В першу чергу за зростанням K3.
2) В другу чергу за спаданням K4.
3) В третю чергу за зростанням K1.
4) В четверту чергу за зростанням K2.

Якщо є заняття з K1 = 1, вони між собою упорядковуються наступним чином:

1) В першу чергу за зростанням K2.
2) В другу чергу за зростанням K3.
3) В третю чергу за спаданням K4.

Якщо немає занять з K1 = 1, та з K3<K3_0 заняття у списку упорядковуються наступним чином:

1) В першу чергу за зростанням K1
2) В другу чергу за зростанням K2.
3) В третю чергу за зростанням K3.
4) В четверту чергу за спаданням K4.

У розклад ставиться заняття, яке опинилось першим у списку. Для цього спочатку з усіх можливих для постановки заняття у розклад навчальних пар розраховується пара з максимальним пріоритетом (розділ 6). Після знаходження навчальної пари з усіх вільних допустимих для проведення заняття аудиторій вибирається аудиторія, яка для цього заняття має найбільший пріоритет. Після цього заняття ставиться у розклад: у відношення STT додається кортеж з атрибутами id_st, id_pare, id_aud, які відповідають поточному заняттю та навчальній парі і аудиторії, що вибрані для нього.

Після постановки заняття у розклад для нього у відношенні Study (ST) виконується відмітка, що заняття поставлене у розклад. З похідних відношень, що містять дані про вільні пари груп, викладачів, аудиторій – FPG, FPT, FPA – видаляються кортежі, в яких ідентифікатор навчальної пари відповідає парі, на яку поставлене заняття. Після цього починається наступна ітерація постановки занять у розклад. Програма припиняє роботу, коли поставлена у розклад вся задана множина занять, або якщо виникла тупикова ситуація – немає жодної пари, на яку можна поставити заняття, тобто для цього заняття K1 = 0.

Якщо в програму додати алгоритм вирішення тупикової ситуації, то програма буде припиняти роботу тільки у випадках, коли дійсно неможливо скласти розклад занять при наявних вихідних даних, наприклад, коли кількість занять, які потребують для проведення однієї той самої аудиторії, більше кількості навчальних пар у тижні. Інші тупикові ситуації, які можна вирішити шляхом переносу деяких занять в іншу аудиторії та/або на іншу навчальну пару, будуть вирішені без припинення роботи програми.

При використанні ітераційного алгоритму складання розкладу занять оптимальність розташування поточної групи занять у розклад можна забезпечити шляхом обліку «коефіцієнта комфортності» навчальної пари, на яку ставиться заняття. В цей «коефіцієнт комфортності» можна включати різноманітні вимоги до оптимального розкладу занять з точки зору студентів і викладачів. Бажано розраховувати «коефіцієнт комфортності» кожної допустимої для заняття навчальної пари і ставити заняття у розклад на пару з максимальним «коефіцієнт комфортності».

Якщо для задачі у якості вихідних даних вибрані досить жорсткі обмеження: невелика кількість допустимих аудиторій для занять або невелика кількість можливих навчальних пар для роботи викладачів, – то програма може закінчити свою роботу не успішно, а по факту виникнення тупикової ситуації. У цьому випадку потрібно, як правило, задати менш жорсткі вихідні дані для проблемних занять, та продовжити роботу програми.

5.6 Урахування занять через тиждень у алгоритмі складання розкладу занять

Для опису занять через тиждень у базі даних додамо у відношення Study атрибут ew (every week), який має значення 1 для занять щотижня, та має значення 0 для занять через тиждень. У таблиці вільних пар груп, викладачів, аудиторій додамо атрибут weeks, що має 3 значення:

NULL – якщо пара вільна щотижня;

0 – якщо пара вільна на непарному тижні;

1 – якщо пара вільна на парному тижні.

Замість відношення Temp3 потрібно знаходити відношення St0 з атрибутами id_st, id_g, id_sub, id_kind, id_t, is_tt, id_a, id_p, ew, weeks, яке містить усі варіанти розташування занять у тримірному просторі (навчальна пара, аудиторія, тиждень). Відношення St0 містить варіанти розташування

щотижневих занять, варіанти розташування занять через тиждень на парному тижні та варіанти розташування занять через тиждень на непарному тижні. Варіанти розташування щотижневих занять містяться у відношенні T3:

$$T1 \leftarrow \sigma_{\text{weeks is NULL}}(FPG) \bowtie \sigma_{\text{weeks is NULL}}(FPT) \quad (5.9)$$

$$T2 \leftarrow \sigma_{ew=1}(Study \bowtie SA) \bowtie \sigma_{\text{weeks is null}}(FPT) \quad (5.10)$$

$$T3 \leftarrow Temp10 \bowtie Temp11 \quad (5.11)$$

Варіанти розташування занять через тиждень на парному тижні містяться у відношенні Temp6:

$$T4 \leftarrow \sigma_{F1}(FPG) \bowtie \sigma_{F1}(FPT) \bowtie \sigma_{weeks=1}(P) \quad (5.12)$$

$$T5 \leftarrow \sigma_{ew=0}(Study \bowtie SA) \bowtie \sigma_{F1}(FPT) \quad (5.13)$$

$$T6 \leftarrow Temp13 \bowtie_{F2} Temp14 \quad (5.14)$$

Варіанти розташування занять через тиждень на непарному тижні містяться у відношенні Temp9:

$$T7 \leftarrow \sigma_{F3}(FPG) \bowtie \sigma_{F3}(FPT) \bowtie \sigma_{weeks=1}(P) \quad (5.15)$$

$$T8 \leftarrow \sigma_{ew=1}(Study \bowtie SA) \bowtie \sigma_{F3}(FPT) \quad (5.16)$$

$$T9 \leftarrow Temp16 \bowtie_{F4} Temp17 \quad (5.17)$$

$$St0 \leftarrow Temp3 \cup Temp6 \cup Temp9 \quad (5.18)$$

У формулах (5.12) – (5.17) предикати F1, F2, F3, F4 розраховуються за формулами (5.19) – (5.22):

$$F1 = (\text{weeks is NULL}) \text{ or } (\text{weeks} = 1) \quad (5.19)$$

$$F2 = ((T5.\text{weeks is NULL}) \text{ or } (T5.\text{weeks} = 1)) \text{ and } T4.\text{weeks}=1 \quad (5.20)$$

$$F3 = (weeks\ is\ NULL\ )\ or\ (weeks = 0) \quad (5.21)$$

$$F4 = ((T5.weeks\ is\ NULL\ )\ or\ (T5.weeks = 0)\ )\ and\ T4.weeks=0 \quad (5.22)$$

Для занять через тиждень отримуємо два кортежі відношення St0 для однієї пари в тій самій аудиторії, якщо пара вільна щотижня для групи, викладача, аудиторії. Таким чином заняття через тиждень будуть мати більшу свободу розташування у розкладі занять.

При вибору для поточного заняття навчальної пари, аудиторії та тижня, воно ставиться у розклад занять – у відношення STT (заняття, що поставлені у розклад) додається кортеж з відповідними значеннями атрибутів.

Відношення STT також має атрибут weeks. Для щотижневих занять цей атрибут призначається NULL, а для занять через тиждень відповідає вибраному тижні для проведення заняття.

Після постановки заняття через тиждень у розклад з похідних відношень, що містять дані про вільні пари груп, викладачів, аудиторій – FPG, FPT, FPA – видаляються кортежі, в яких ідентифікатор навчальної пари відповідає парі, на яку поставлене заняття тільки у випадку, коли ця пара у відношенні вільна тільки на цьому тижні. Якщо вибрана пара вільна щотижня, кортеж з відношення не усувається, а в ньому змінюється значення атрибуту weeks зі значення NULL на значення, що відповідає протилежному тижню.

6 АЛГОРИТМ ВИЗНАЧЕННЯ ОПТИМАЛЬНОЇ НАВЧАЛЬНОЇ ПАРИ ДЛЯ ПРОВЕДЕННЯ ЗАНЯТТЯ

6.1 Розрахунок пріоритетів вільних навчальних пар для постановки заняття у розклад

Алгоритм додавання заняття до розкладу повинен передбачати виконання декількох умов, наприклад:

1) в першу чергу заняття ставиться на пару, що є «вікном» у розкладі;
2) якщо «вікон» у допустимої для даного заняття множині пар немає, то заняття ставиться поряд (на попередній або наступній парі) з заняттям, що вже є у розкладі;
3) якщо на допустимій для даного заняття множині навчальних пар не виконуються умови 1 та 2, заняття ставиться у вільний від занять день на мінімально можливу пару.
4) якщо на допустимій для даного заняття множині навчальних пар не виконуються умови 1 – 3, заняття ставиться зі створенням «вікна» у розкладі занять

Розрахунок «вікон» у розкладі занять групи або викладача є простою задачею при використанні реляційних баз даних с їх багатою мовою запитів – мовою SQL [6, 15–16].

Множина зайнятих заняттями пар в реляційній базі даних відображається у таблицю-відношення. При виборці для деякої академічної групи зайнятих навчальних пар з групуванням по днях тижня та розрахунком мінімальної, максимальної зайнятої пари и кількості зайнятих пар у кожен день тижня, легко визначити «вікно» у розкладі занять студентов. «Вікно» у розкладі занять є в той день, коли:

$$P_{max} \geq P_{min} + K, \qquad (6.1)$$

де P_{max} – номер останньої зайнятої пари в деякий день тижня;

P_{min} – номер першої зайнятої пари в цей день тижня;

K – кількість зайнятих пар протягом цього дня.

Всім вільним парам потрібно призначити пріоритети, у відповідності до яких слід їх вибирати для постановки заняття у розклад. При визначення пріоритетів вільних навчальних пар слід вважати оптимальним розклад

занять академічних груп з чотирма навчальними днями тижня, в кожному з яких зайняті перші три пари.

З урахуванням чотирьох умов додавання занять до розкладу, що перераховані вище, можна призначити наступні пріоритети для постановки нового заняття у розклад (рис.6.1):

1) максимальний пріоритет (1) призначається парі, яка є «вікном» у розкладі занять групи;
2) наступний пріоритет (2) призначається другій парі для дня, коли є тільки перша пара; також цей пріоритет призначається для пари з номером Pmin – 1, якщо в цей день є тільки одна пара у розкладі, і ця пара не перша;
3) наступний пріоритет (3) призначається для третьої пари, якщо в цей день у розкладі є тільки одна друга пара;
4) 4-й пріоритет призначається для четвертої пари, якщо в цей день у розкладі є тільки одна третя пара;
5) 5-й пріоритет призначається для пари з номером Pmin – 1, якщо в цей день є дві пари поспіль – 2 та 3 або 3 та 4;
6) 6-й та 7-й пріоритети призначається для пари з номером Pmax + 1, якщо в цей день є дві пари поспіль – 1 та 2 або 2 та 3;
7) 8-й пріоритет призначається першій парі у день, коли є друга та четверта пари;
8) 9-й та 10-й пріоритети призначаються для перших двох пар у вільний день:
9) Останні 5 пріоритетів призначаються вільним парам у інших випадках в залежності від того, наскільки постановка заняття на ці пари може ускладнити процес створення оптимального розкладу занять.

За вказаним алгоритмом можна розраховувати пріоритети вільних навчальних пар групи та викладача. Але вимоги до розкладу занять студентів є більш жорсткими, ніж вимоги до розкладу занять викладачів. Для розкладу занять академічних груп неприпустимо наявність «вікон», в той час як для розкладу занять викладача ставиться умова наявності мінімальної кількості «вікон». Також для розкладу занять викладача допустима наявність однієї пари занять у деякий день тижня, що є неприйнятним для розкладу занять академічної групи.

1	9		2	11	15
2	10	2		2	11
3	13	10	3		2
4	15	15	14	4	

1		5	14		8
2			5	1	
3	6				1
4	15	7		12	

1			8		
2	1				1
3	1			1	
4		12			

Рисунок 6.1 – Пріоритети для вибору вільної пари для заняття

Таким чином, при визначенні загального пріоритету навчальній пари для постановки заняття у розклад пріоритет цієї пари для академічної групи повинен враховуватись з більшим ваговим коефіцієнтом, ніж пріоритет для викладача.

Вказану умову можна виконати, якщо загальний пріоритет навчальної пари розраховувати за формулою (6.2):

$$PRp = PRg + Wt \cdot PRt, \qquad (6.2)$$

де PRp – загальний пріоритет навчальної пари;

PRg – пріоритет навчальної пари в розкладі занять групи;

PRt – пріоритет навчальної пари в розкладі занять викладача;

Wt – ваговий коефіцієнт, $0 < Wt < 1$.

Коефіцієнт Wt також повинен зберігатись у базі даних в таблиці емпіричних коефіцієнтів, які можна змінювати перед початком розрахунку розкладу занять.

Сумарний пріоритет PRp розраховується для всіх навчальних пар, допустимих для постановки заняття у розклад. Потім вибирається навчальна пара з максимальним пріоритетом, і на цю пару призначається заняття.

6.2 Урахування занять через тиждень у розрахунку пріоритетів вільних навчальних пар

Наявність у розкладі занять через тиждень значно ускладнює алгоритм розрахунку оптимальної навчальної пари для постановки заняття у розклад. У випадку врахування лише щотижневих занять, як видно з рис. 6.1, існує тільки 15 варіантів з різним положенням зайнятих та вільних пар, та серед них 32 варіанти вільних пар, для яких потрібно призначити пріоритет. У випадку урахування занять через тиждень до вказаних варіантів положень зайнятих та вільних пар додаються ще варіанти, зображені на рис. 6.2а – 6.2в.

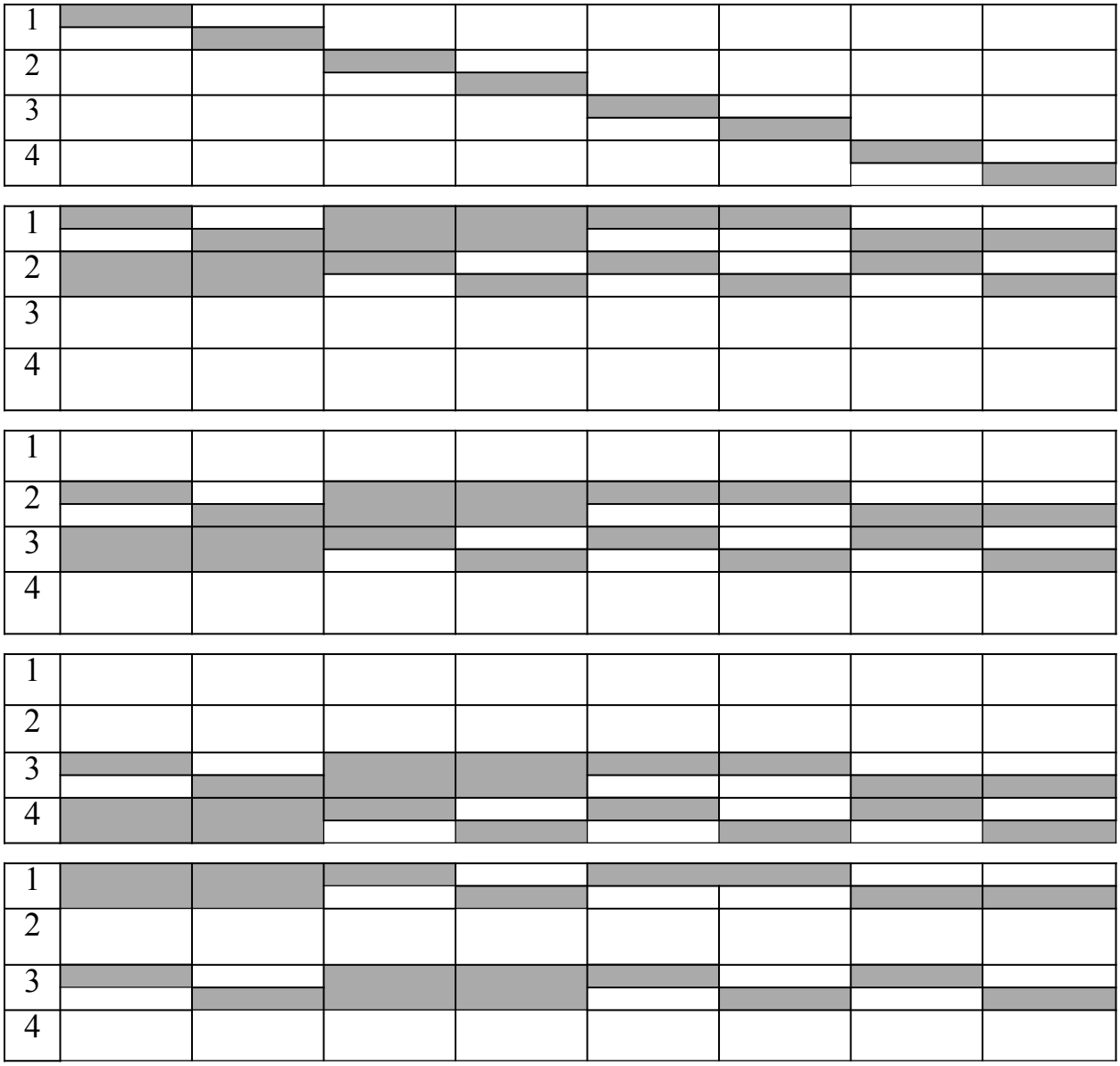

Рисунок 6.2а – Варіанти взаємного розташування зайнятих та вільних навчальних пар при складанні розкладу занять

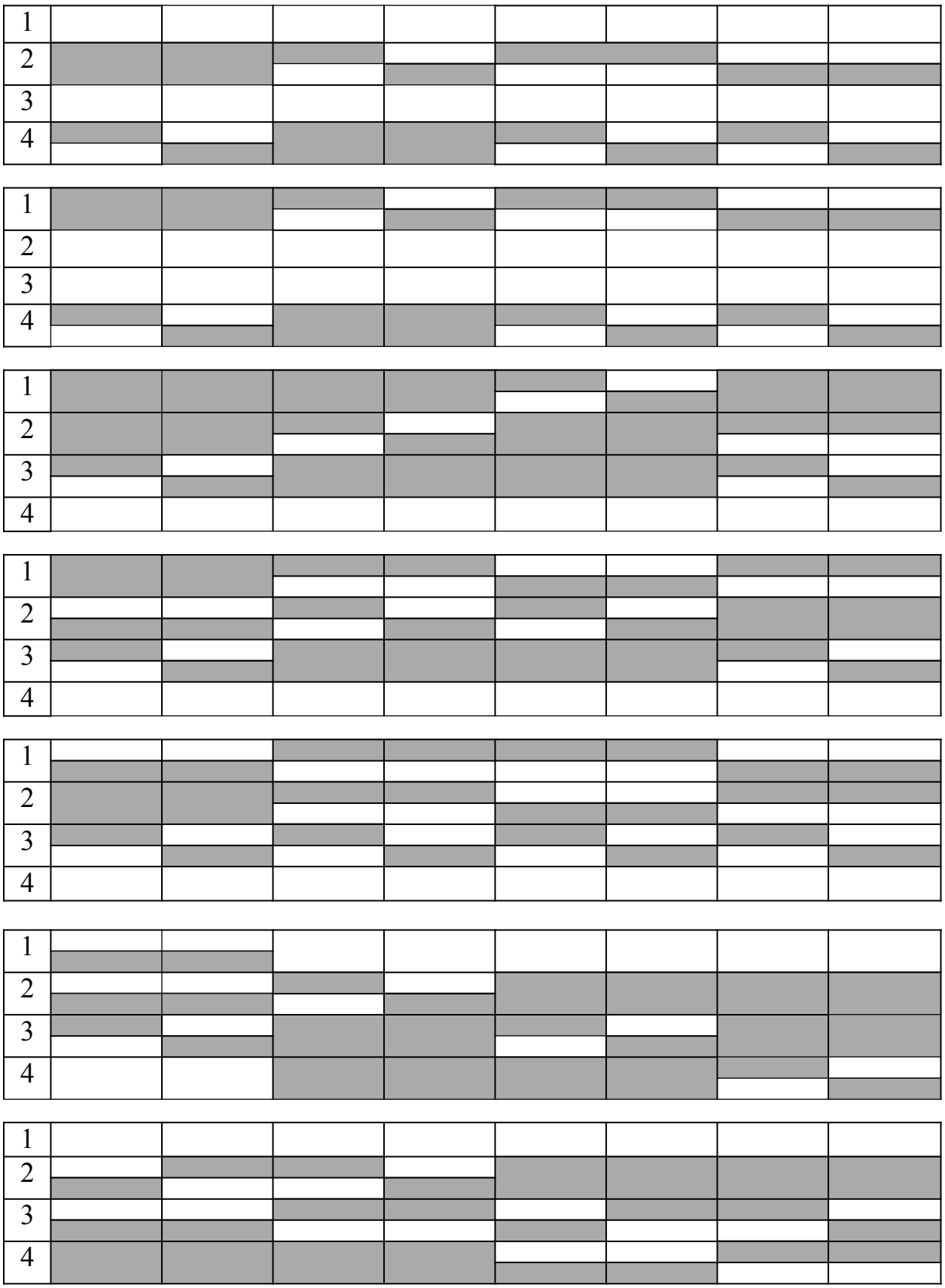

Рисунок 6.2б – Варіанти взаємного розташування зайнятих та вільних навчальних пар при складанні розкладу занять

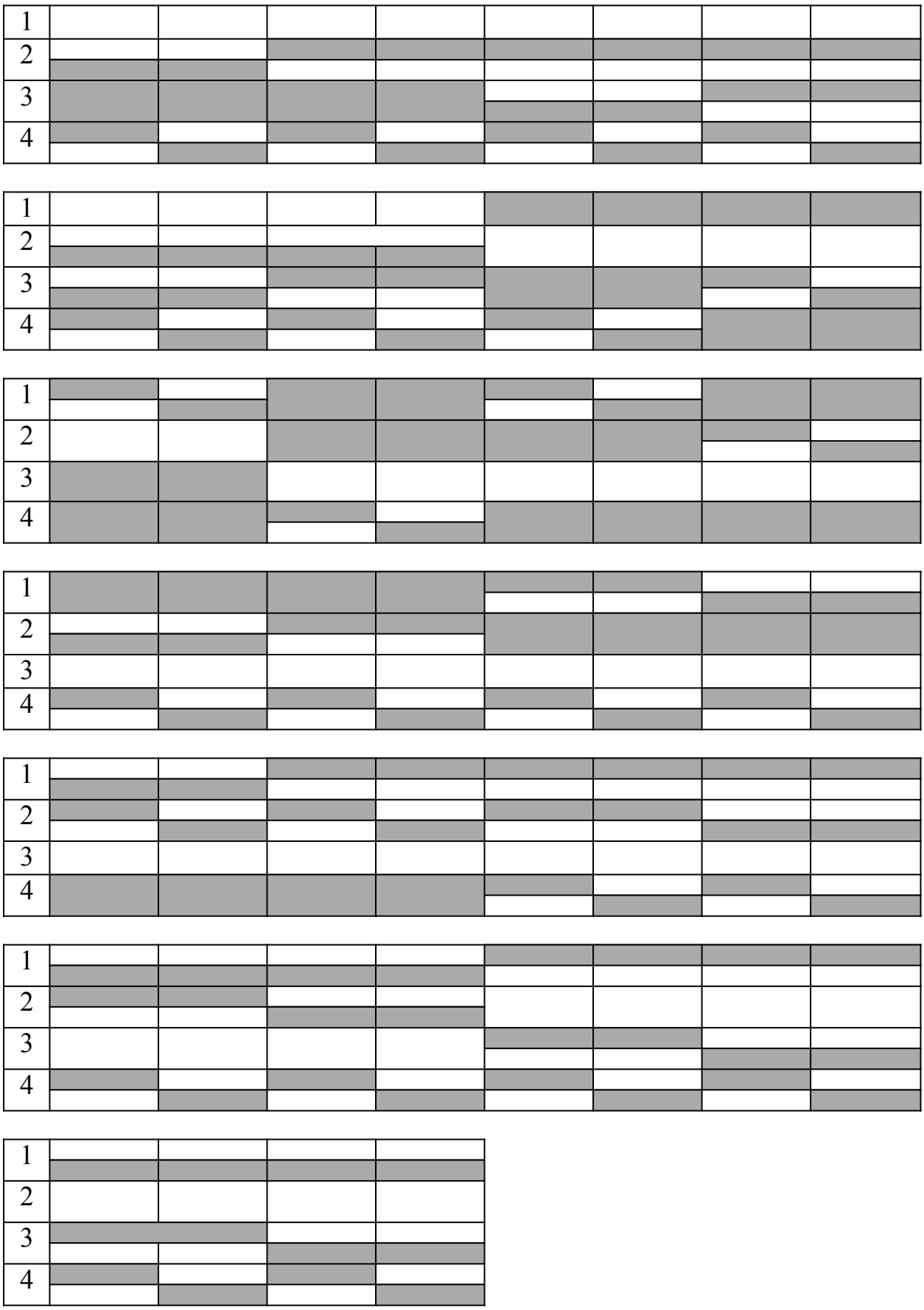

Рисунок 6.2в – Варіанти взаємного розташування зайнятих та вільних навчальних пар при складанні розкладу занять

На рис. 6.1, 6.2а – 6.2в відображені лише варіанти розташувати зайнятих та вільних пар для випадків постановки у розклад заняття кожного тижня. Ці заняття можуть ставитись тільки на пару, яка є вільною кожного тижня.

У випадках постановки у розклад заняття через тиждень крім вказаних на рис. 6.1, 6.2а – 6.2в варіантів постановки заняття додаються ще 80 варіантів розкладу занять продовж дня з чотирма парами, з яких від однієї до чотирьох пар можуть бути зайняті через тиждень з різними поєднаннями парних та непарних тижнів. Для варіантів розкладів занять, вказаних на рисунках, заняття через тиждень може ставиться як на пару, що вільна через тиждень, так і на пару, що вільна кожен тиждень. Якщо пара вільна кожного тижня, то заняття через тиждень може ставиться або на парний тиждень, або на непарний тиждень.

При призначенні вільним навчальним парам пріоритетів для постановки заняття в розклад потрібно виходити з того, наскільки постановка даного заняття на дану пару покращує або погіршує розклад, якщо оптимальним вважати розклад занять академічних груп з трьома першими парами занять в день протягом 4 днів (для 19 – 24 академічних годин занять на тиждень).

6.3 Відступ від послідовного складання розкладу занять

У деяких випадках для запобігання появи великої кількості «вікон» у розкладі одного тижня за рахунок постановки заняття на інший тиждень бажано ставити одночасно на вибрану вільну щотижня пару два заняття через тиждень: одне заняття на парний тиждень, інше – на непарний. Подібні випадки постановки занять зображені на рис. 6.3.

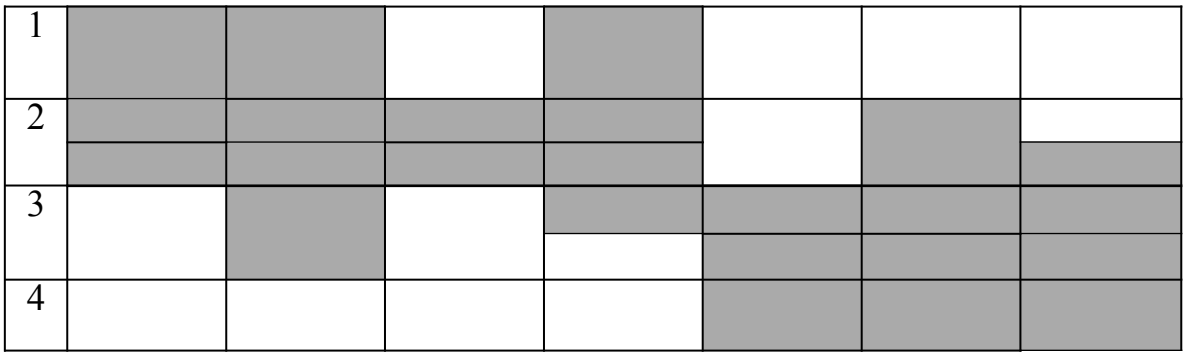

Рисунок 6.3 – Випадки одночасної постановки у розклад двох занять

На рис. 6.3 діагональною штриховкою відмічені зайняті пари у розкладі занять групи; вертикальною штриховкою відмічені пари, на які ставиться поточне заняття у списку; горизонтальною штриховкою відмічені пари, на які бажано поставити якесь заняття разом з поточним заняттям.

Пошук занять, які можна поставити на вибрану пару на іншому тижні, виконується вибіркою занять, для яких ця пара є допустимою для постановки у розклад. Вибірка виконується з тимчасової таблиці St0 (формула 5.3) у тимчасову таблицю TT з тими ж атрибутами. Нехай поточне заняття з ідентифікатором id_st = I у групі з ідентифікатором id_g = J ставиться на пару з ідентифікатором id_pare = N на тиждень з позначкою weeks =W. Потрібно знайти всі заняття цієї ж групи, що проводяться через тиждень і мають серед доступних для постановки у розклад вибрану пару на протилежному тижні:

$$TT \leftarrow \sigma_{id\_st\ != I\ \wedge\ id\_g = J\ \wedge\ id\_p = N\ \wedge\ weeks = !W}\ (\ St0) \qquad (6.3)$$

Нехай поточне заняття проводить викладач з ідентифікатором id_t = K. З відношення TT вибирається підмножина занять цього викладача – відношення TT0:

$$TT0 \leftarrow \sigma_{id\_t = K}\ (\ TT) \qquad (6.4)$$

Якщо відношення TT0 не порожнє, то з нього вибирається заняття з мінімальною свободою розташування у розкладі. Це заняття і ставиться у розклад.

Якщо відношення TT0 порожнє, для всіх викладачів, що ведуть заняття з відношення TT0 розраховується пріоритет навчальної пари id_pare = N, weeks = !W. Вибирається заняття, яке має самий високий пріоритет викладача на вказаній парі.

Обидва заняття ставляться у розклад.

7 СИСТЕМНИЙ АНАЛІЗ І КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Для використання програми автоматизованого складання розкладу занять потрібно розробити базу даних інформаційної системи «Наавчальний процес». Необхідність використання бази даних випливає з великого обсягу даних, які використовуються при вирішенні задачі складання розкладу. Крім того, як було сказано в попередніх розділах, задача складання розкладу добре формулюється в термінах теорії відношень та реляційної алгебри. Реляційна алгебра і реляційне числення лежать в основі стандартної мови реляційних СУБД – мови SQL. Отже, логічно припустити, що рішення задачі складання розкладу занять у ЗВО можна шукати програмними засобами реляційних СУБД.

7.1 Етапи проектування бази даних

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

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

Основні етапи концептуального проектування [6, 17]:

- 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 транзакціях користувачів;
- обговорення локальних концептуальних моделей даних з кінцевими користувачами.

Завданням логічного проектування бази даних є створення логічної моделі на основі обраної моделі даних, але без урахування конкретної СУБД, яка буде використовуватися, і інших фізичних аспектів реалізації

інформаційної системи. На етапі логічного проектування отримана концептуальна модель уточнюється і перетворюється для відповідності структурам даних і зв'язків між ними, властивих обраної моделі даних: реляційної, об'єктно-орієнтованої, об'єктно-реляційної, і т.д.

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

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

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

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

7.2 Концептуальне проектування бази даних

В університеті кафедри відносяться до факультетам; дисципліни і викладачі приписані до кафедр; академічні групи відносяться до факультетам та курсам; кожен група вчиться за своїми навчальними планами, зі своїми дисциплінами. За дисциплінами можуть передбачатися заняття різного виду: лекції, практичні заняття (семінари), лабораторні заняття. Таким чином, є

такі основні типи сутностей в базі даних: Факультет, Кафедра, Група, Викладач, Предмет (навчальна дисципліна), Вид_заняття.

Між основними типами сутностей існують такі типи зв'язків: факультет має (містить) групу, зв'язок між типами сутностей Факультет Група «один до багатьох»; кафедра має (містить) викладачів, тип зв'язку між типами сутностей Кафедра і Викладач «один до багатьох»; до кафедри приписані предмети (навчальні дисципліни), тип зв'язку між типами сутностей Кафедра і Предмет «один до багатьох»; у групи є навчальний план на семестр, тип зв'язку між типами сутностей Навчальний_план та Група «один до багатьох» (в межах семестру).

Кожен пункт навчального плану описує навантаження студентів по одній навчальній дисципліні (предмету). Тому необхідна похідна сутність Предмет_плана. Тип сутності Предмет_плана пов'язаний з типами сутностей Навчальний_план, Вид_занять і Предмет. Тип зв'язку між типами сутностей Навчальний_план і Предмет_плана буде «один до багатьох». Тип зв'язку між типами сутностей Предмет і Предмет_плана буде «один до одного» для навчального плану однієї групи в одному семестрі; в загальному випадку тип зв'язку між типами сутностей Предмет і Предмет_плана буде «один до багатьох», оскільки одну і ту ж дисципліну часто вивчають студенти, які навчаються за різними навчальними планами. Між типами сутностей Вид_занять і Заняття буде тип зв'язку «один до багатьох».

При складанні розкладу необхідна похідна сутність Заняття: одна навчальна пара з одного предмета. Тип сутності Предмет_плана пов'язаний з типом сутності Заняття по типу зв'язку «один до багатьох».

Отримуємо першу ER-діаграму бази даних (рис.7.1).

Діаграма на рис. 7.1 має деяку неточність. Навчальний план приписаний не до академічної групі, а до факультету. Між типами сутностей Факультет і Навчальний_план існує тип зв'язку «один до багатьох».

Академічні групи навчаються за навчальними планами, які є на їх факультеті, при цьому протягом навчального року кілька академічних груп можуть вчитися по одному і тому ж навчальному плану. Тобто між типами сутностей Навчальний_план і Група існує тип зв'язку «один до багатьох».

Навчальні плани факультетів складаються на весь навчальний рік, а розклад занять – на один семестр.

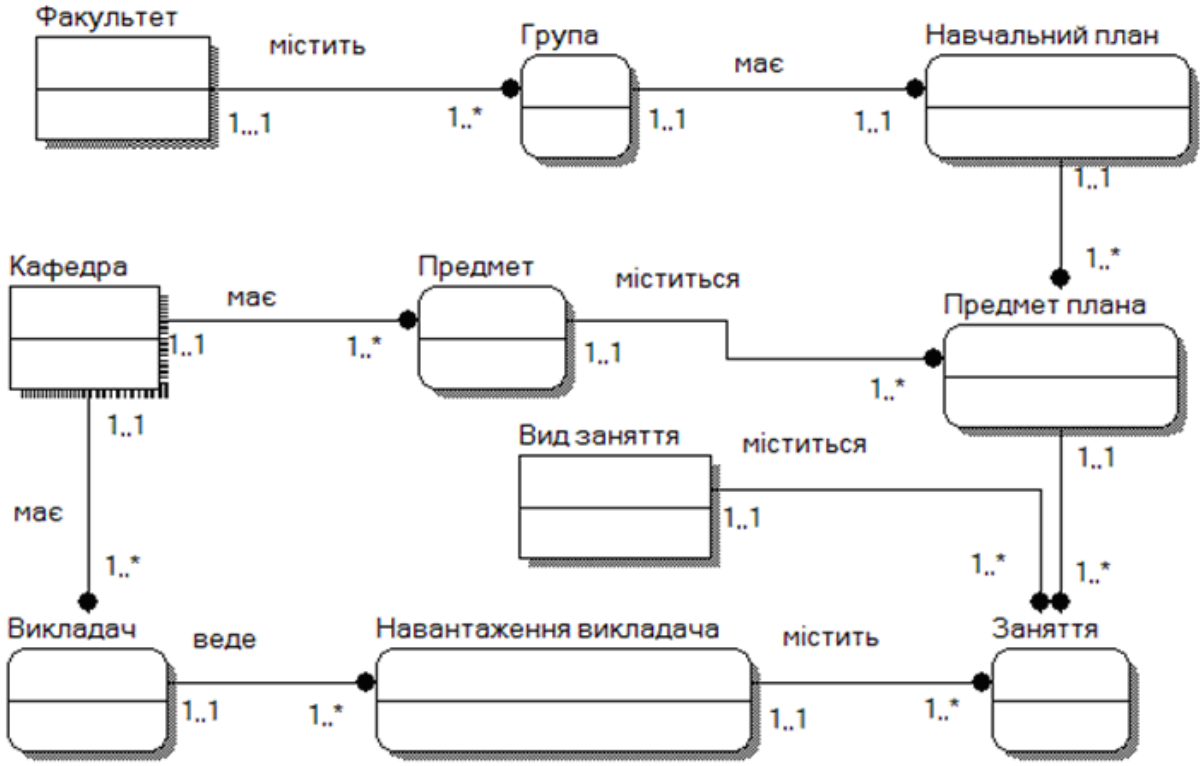

Рисунок 7.1 – Перша ER-діаграма бази даних «Розклад занять»

Реальний навчальний план на навчальний рік повинен відображатися на два навчальних плани на семестр. Тому тип сутності Навчальний_план буде пов'язаний з типом сутності Група по типу зв'язку «багато до багатьох».

Отримуємо другу ER-діаграму бази даних «Розклад занять» (рис. 7.2).

Кожне заняття групи повинно бути приписано до якоїсь з аудиторій. Отримуємо нову основний тип сутності Аудиторія. До постановки в розклад тип зв'язку між типами сутностей Заняття_групи і Аудиторія в загальному випадку буде «багато до багатьох», оскільки в кожній аудиторії можуть проводитися багато занять, а кожне заняття може бути призначено в одну з декількох відповідних аудиторій.

Отримуємо доповнену ER-діаграму концептуальної моделі БД «Розклад занять» (рис. 7.3).

ER-діаграма на рис. 7.3 відображає типи сутностей, які описують підсхему «Вихідні дані». Як описано в розділі 5, при складанні розкладу занять з'являються нові типи сутностей: Навчальна_пара, Вільна_пара_групи, Вільна_пара_викладача, Вільна_пара_аудіторії.

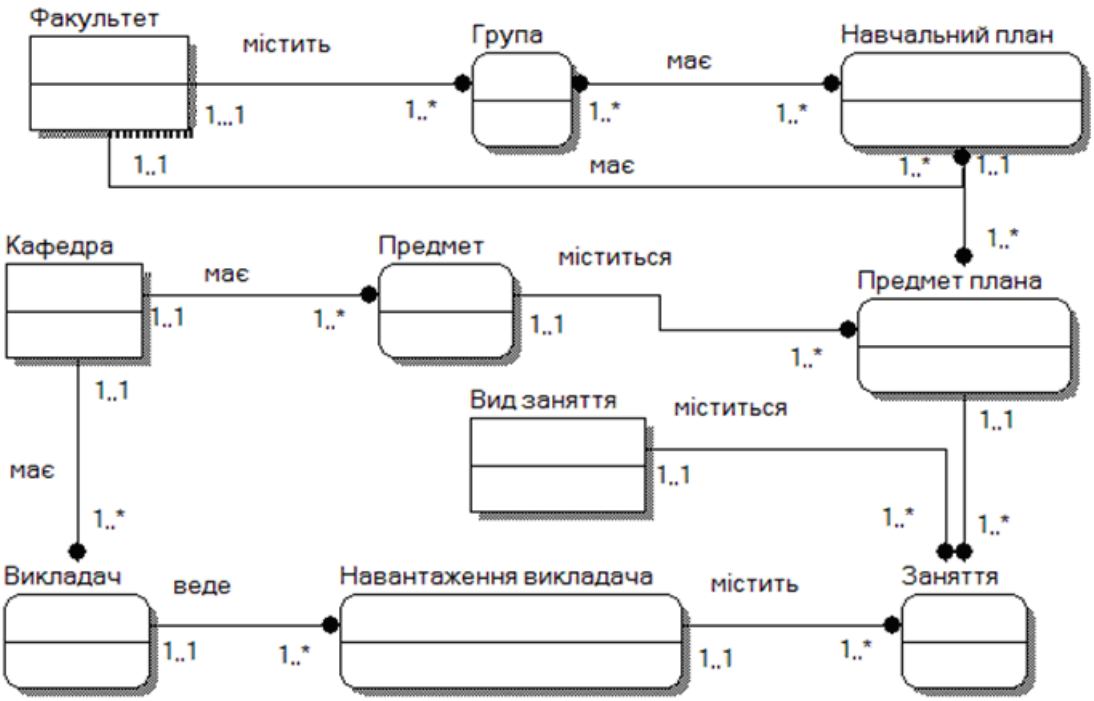

Рисунок 7.2 – Друга ER-діаграма бази даних «Розклад занять»

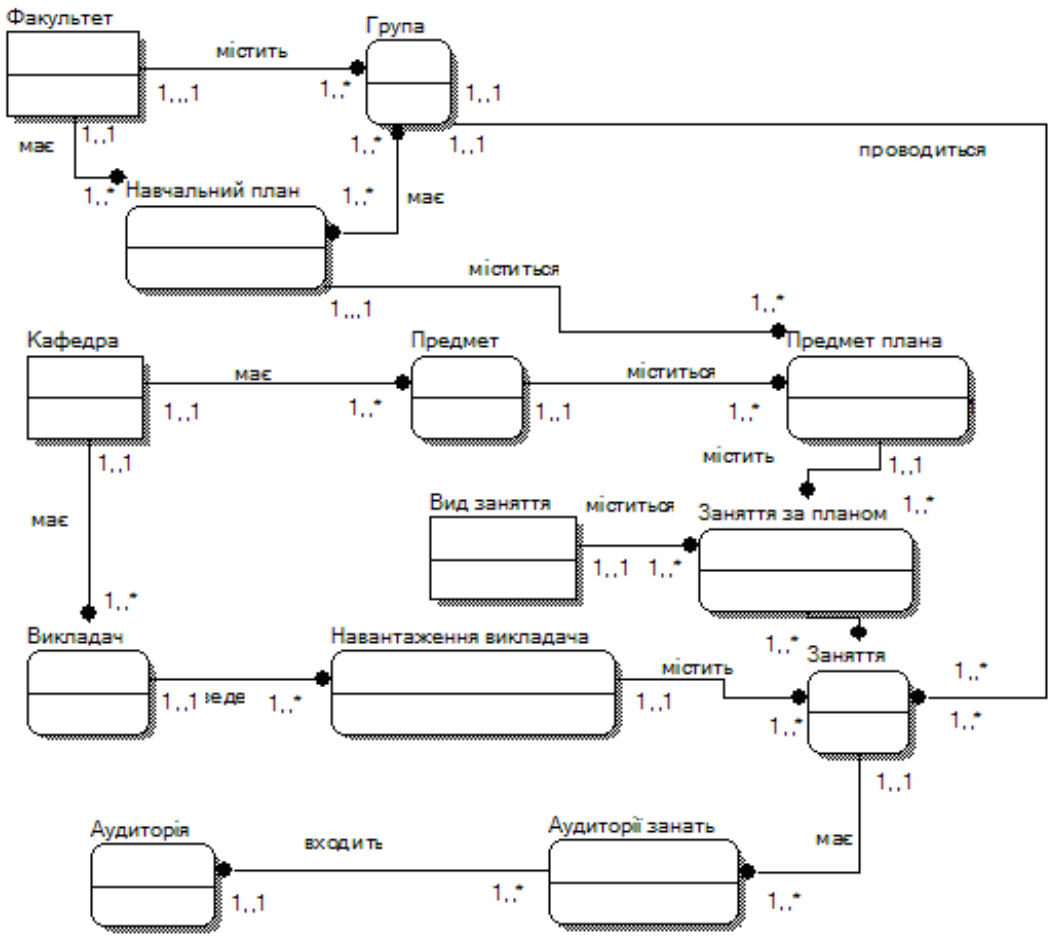

Рисунок 7.3 – Доповнена ER-діаграма БД «Розклад занять»

Як видно з формул 5.1 – 5.3, для знаходження допустимої навчальної пари для постановки заняття в розклад необхідні тільки типи сутностей Заняття, Аудиторія_занять, Вільна_пара_викладача, Вільна_пара_групи, Вільна_пара_аудіторії. Для визначення найбільш зручною для постановки в розклад пари для заняття потрібні ще типи сутностей Зайнята_пара_групи і Зайнята_пара_викладача, які використовуються при розрахунку пріоритетів вільних пар з точки зору оптимальності постановки заняття в розклад академічної групи та розклад викладача. Тому в підсхемі «Розрахунок розкладу» інші типи сутностей можна опустити (рис. 7.4).

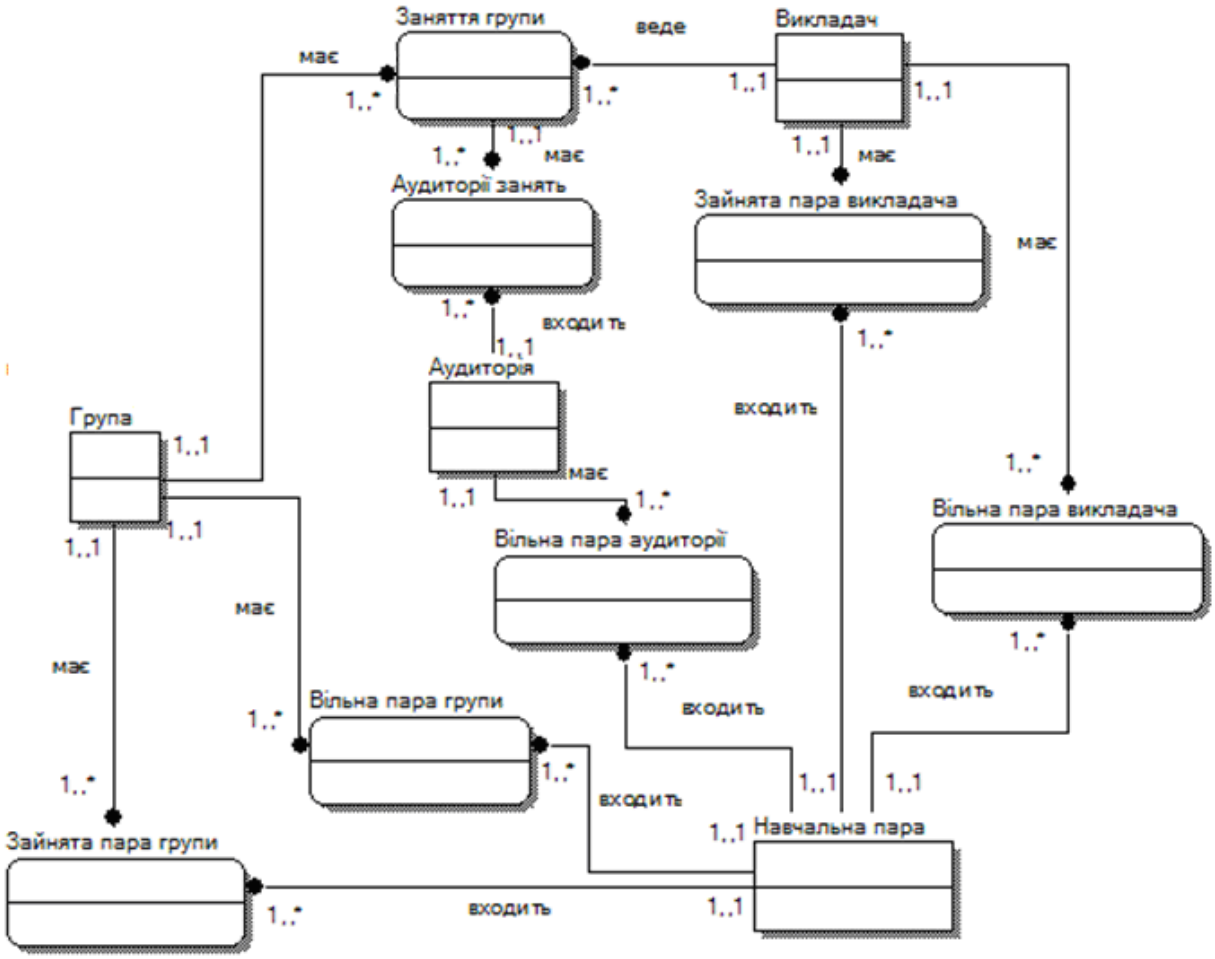

Рисунок 7.4 – ER-діаграма БД «Розклад занять»:
підсхема «Розрахунок розкладу»

Останній тип сутності, який бажано ввести в схему, це похідний тип сутності Заняття_у_розкладі. Цей тип сутності необов'язковий, можна в тип сутності Заняття_групи додати атрибути, які описують навчальну пару, на

яку поставлено заняття в розкладі, і аудиторію, в яку це заняття призначено. До постановки заняття в розклад ці атрибути повинні мати пусте значення – NULL. Після постановки в розклад вони не можуть мати пусте значення.

Опис подібного виду атрибутів незручний, тому краще додати тип сутності Заняття_в_розкладі. З цим типом сутності матимуть тип зв'язку «один до багатьох» наступні типи сутностей: Аудиторія і Навчальна_пара. Між типами сутностей Заняття і Заняття_в_розкладі буде тип зв'язку «один до одного».

Похідні відношення Зайнята_пара_групи та Зайнята_пара_викладача при наявності типу сутності Заняття_в_розкладі будуть тимчасовими або віртуальними таблицями бази даних, які отримують необхідні дані операцією вибірки зі зв’язаних таблиць Заняття і Заняття_в_розкладі.

8 ВИБІР СУБД ТА ЗАСОБІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для створення бази даних «Розклад занять» як складової частини інформаційної системи університету потрібно визначитись з системою управління базами даних, яка вибирається для створення БД серверного ПЗ для неї.

8.1 Порівняння MySQL Server та MS SQL Server

MySQL – вільна система управління базами даних. Розробка та підтримка сайта MySQL здійснює корпорація Oracle, що має на даний момент права на торговельну марку. Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією. Крім цього розробники створюють функціональність за замовленням ліцензійних користувачів, саме завдяки такому замовленню майже в найраніших версіях з'явився механізм реплікації.

MySQL є рішенням для малих і середніх додатків. Входить до складу серверів WAMP, AppServ, LAMP і в портативні збірки серверів Денвер, XAMPP. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми [18].

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць.

MySQL є найбільш пристосованою для застосування в середовищі веб системою управління базами даних. При цьому вона стала непорушним стандартом в області СУБД для веб, в якій розвиваються можливості для використання її в будь-яких критичних бізнес-додатках, що створює конкуренцію на рівних з СУБД таких виробників, як Oracle, IBM, Microsoft і Sybase.

Основні переваги MySQL:

- многопоточность, підтримка декількох одночасних запитів;

- оптимізація зв'язків з приєднанням кількох даних за один прохід;
- записи фіксованої і змінної довжини;
- ODBC драйвер;
- гнучка система привілеїв і паролів;
- гнучка підтримка форматів чисел, рядків змінної довжини і міток часу;
- інтерфейс з мовами C і Perl, PHP;
- швидка робота, масштабованість;
- сумісність з ANSI SQL;
- безкоштовна в більшості випадків;
- хороша підтримка з боку провайдерів послуг хостингу;
- швидка підтримка транзакцій через механізм InnoDB.

Типи таблиць, які підтримуються у MySQL:

- MyISAM
- InnoDB
- Berkeley
- Merge
- Heap.

Microsoft SQL Server – система керування базами даних (РСУБД), розроблена корпорацією Microsoft. Основний використовуваний мову запитів – Transact-SQL (T-SQL), створений спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO по структурованого мови запитів (SQL) з розширеннями. Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства; конкурує з іншими СУБД в цьому сегменті ринку [16, 19, 20].

Ядро реляційної бази даних забезпечує безпечні, надійні, масштабовані операції над реляційними даними та дозволяє працювати як зі структурованими, так і з неструктурованими XML-даними. Ядро бази даних забезпечує підтримку .NET CLR (можливість створення збережених процедур, функцій і тригерів на керованому коді, а також визначаються користувачем типів і агрегатних функцій) і розширень ADO.

Служби реплікації забезпечують реплікацію даних між різними базами даних при створенні розподілених і мобільних додатків. Ці служби можуть використовуватися в якості джерел даних для створення звітів, а також підтримують інтеграцію з гетерогенними системами, включаючи бази даних, керовані СУБД Oracle.

Висока доступність і відмовостійкість

Під високою доступністю розуміється забезпечення роботи СУБД в режимі 24/7 з підтримкою операцій створення резервних копій, робота в кластерах, підтримка віддзеркалення баз даних, онлайнові операції (індексація, відновлення і реакція на апаратні зміни), а також мінімальні витрати на відновлення після збоїв (база даних доступна при виконанні операцій відкату).

Керованість

Під керованістю розуміється підтримка операцій, пов'язаних з автоматичним настроюванням для забезпечення його оптимальної роботи, наявність засобів управління, засобів напівавтоматичного налаштування, засобів отримання звітності про роботу бази даних, включаючи можливість побудови звітів, забезпечення повнотекстового пошуку, а також наявність служб створення розкладів для виконання певних робіт .

Безпека

До характеристик безпеки відносяться підтримка аутентифікації, авторизації, ведення протоколу (аудит), підтримка шифрування даних і управління ключами.

8.2 Вибір СУБД

З порівняння СУБД MySQL та MS SQL Server видно, що кожна з них має свої переваги. При розробці веб-додатків більш орієнтуються на MySQL.

Інформаційна система університету повинна працювати також з офісними додатками, які частіше розробляють для СУБД MS SQL Server та Oracle. Інформаційна система розробляється поступово протягом декількох останніх років на базі СУБД MS SQL Server 2005. Ця версія СУБД є безкоштовною, для неї вже розроблена база даних та набір збережених процедур для одробки даних. Керуючись тим, що одним з головних вимог при створенні автоматизованої системи розкладу занять є інтеграція в єдиний інформаційний простір всіх розроблених ІС, що мають відношення до навчального процесу університету, було прийнято рішення про використання в основі розроблюваної системи СУБД MS SQL Server 2005.

9 ОПИС СЕРВЕРНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Алгоритм складання розкладу занять ґрунтується на алгебрі відношень, тому програмна реалізація його можлива лише засобами мови запитів SQL. Процедури складання розкладу занять повинні бути реалізовані у вигляді збережених процедур бази даних [15, 16, 19, 20].

Адміністратор БД може запускати процес автоматизованого складання розкладу занять на сервері, але для нормальної роботи з програмою потрібно розробити клієнтський додаток – інтерфейсную частину програми складання розкладу занять.

Користувач, наприклад, диспетчер навчального відділу, за допомогою клієнтського додатку обирає необхідні параметри роботи програми і запускає процедуру розрахунку розкладу для заданої підмножини занять.

Якщо програма завершилась успішно, користувач може переглянути готовий розклад занять, а також відкоригувати розклад занять академічних груп засобами програми «АРМ диспетчера навчального відділу». Потім він обирає наступну множину академічних груп для постановки у розклад їх занять. Після постановки у розклад занять всіх академічних груп ЗВО потрібно переглянути розклад занять викладачів та за потребою відкоригувати розклад.

Для перегляду розкладу занять академічних груп та викладачів у базі даних створюються представлення користувачів, які дозволяють виводити розклад занять у зручному для користувача вигляді. Ці представлення також входять до складу серверного програмного забезпечення бази даних «Розклад занять».

9.1 Опис збережених процедур БД «Розклад занять»

Процедура MakeTimeTable ставить у розклад задану множину занять:

```
CREATE proc MakeTimeTable
@k1 tinyint,
@k2 tinyint,
@kp_st tinyint,
@TeachGr numeric(3,2),
@nMax int,
@id_st int output,
@k tinyint output,
```

@msg varchar(200) output

Параметри процедури:

@k1 – параметр K1MAX в формулах розрахунку коефіцієнтів для ранжирування списку занять при постановці у розклад (розділ 5.3);

@k2 – параметр K2MAX в формулах розрахунку коефіцієнтів для ранжирування списку занять при постановці у розклад (розділ 5.3);

@kp_st – параметр K3MAX в формулах розрахунку коефіцієнтів для ранжирування списку занять при постановці у розклад (розділ 5.4);

@TeachGr – коефіцієнт Wt у формулі (6.2);

@nMax – максимальна кількість занять, яку потрібно поставити у розклад при даному виклику процедури;

@id_st – ідентифікатор заняття, на якому процедура припинила роботу через виникнення тупикової ситуації;

@k – код закінчення роботи процедури, якщо процедура успішно завершила роботу, то @k = 0;

@msg – текст повідомлення про результат роботи процедури.

Процедура MakeTimeTable знаходить кількість викладачів з критично низькою свободою розташування їх занять у розклад. Якщо таких викладачів не знайдено, список занять для постановки у розклад упорядковується спочатку за коефіцієнтами K1, K2, потім – за коефіцієнтами K3, K4 (розділ 5.5). Якщо Вказані викладачі існують, список занять для постановки у розклад упорядковується спочатку за коефіцієнтами K3, K4, потім – за коефіцієнтами K1, K2.

Для постановки у розклад вибирається заняття, що опинилось у голові списку, та знаходяться значення змінних: ідентифікатор цього заняття @id_study; ідентифікатор групи, в якій проводиться заняття, id_gr; ідентифікатор викладача, що веде заняття, @id_teach; параметр @week, що визначається, чи проводиться заняття кожен тиждень.

Для розрахунку пріоритетів вільних пар викладача викликається процедура FillTeachParePriorWeek з параметрами @id_study, @id_teach, @week.

Для розрахунку пріоритетів вільних пар групи викликається процедура FillGrParePriorWeek з параметрами @id_study, @id_gr, @week.

Список вільних пар упорядковується у відповідності до сумарного пріоритету групи та викладача. Для постановки заняття у розклад вибирається навчальна пара з максимальним пріоритетом @id_pare. Вибирається аудиторія @id_aud, вільна на цій парі, з максимальним пріоритетом для проведення цього заняття. Якщо заняття проводиться через тиждень вибирається тиждень для проведення @weeks.

Якщо заняття проводиться кожен тиждень, тобто @week = 1, для постановки заняття у розклад викликається процедура StudyToTTEveryWeek з параметрами @id_study, @id_gr, @id_teach, @id_pare, @id_aud.

Якщо заняття проводиться через тиждень, тобто @week = 0, для постановки заняття у розклад викликається процедура StudyToTTWeek з параметрами @id_study, @id_gr, @id_teach, @id_pare, @id_aud, @weeks.

Процедура MakeTimeTable припиняє роботу з кодом @k = 0, якщо у розклад поставлені @nMax занять, або поставлені усі заняття з відношення Study з атрибутом is_tt = 0 (заняття, що потрібно поставити у розклад).

Процедура припиняє роботу з кодом @k = 1, якщо у розкладі у відношенні Study не було кортежів з атрибутом is_tt = 0, тобто при запуску програми не була задана множина занять, які потрібно поставити у розклад.

Процедура припиняє роботу з кодом @k = 2, якщо виникла тупикова ситуація під час складання розкладу занять. У цьому випадку повертається значення ідентифікатора заняття @id_study, для якого виникла тупикова ситуація.

Процедура FillTeachParePriorWeek розраховує пріоритети вільних пар викладача для постановки заняття у його розклад:

CREATE proc FillTeachParePriorWeek
@id_teach smallint,
@id_study int,
@EveryWeek bit

Параметри процедури:

@id_teach – ідентифікатор викладача;

@id_study – ідентифікатор заняття;

@EveryWeek – позначка, чи проводиться заняття кожного тижня.

Процедура FillTeachParePriorWeek вибіркою з відношень Study, STT заповнює тимчасові таблиці TeachParePrior та TeachParePriorWeek. Таблиця TeachParePrior для кожної навчальної пари тижня містить відмітку, є на цій парі заняття у викладача, чи немає. Таблиця TeachParePriorWeek містить аналогічні дані, але окремо для парного та непарного тижня.

Операцією вибірки з таблиці TeachParePrior з групуванням визначається для кожного дня навчального тижня кількість наявних пар у розкладі занять викладача @@k, номер першої @@Pmin та номер останньої @@Pmax пари протягом дня.

У циклі по днях навчального тижня для кожного дня тижня викликається процедура розрахунку пріоритетів вільних у цей день навчальних пар викладача. У залежності від значення @@k та @EveryWeek викликається одна з процедур призначення вільним парам даного дня пріоритетів для викладача.

Процедура DoPriorTeachWeek_2 розраховує пріоритети вільних пар викладача для випадку постановки у розклад щотижневого заняття. Процедура розраховує пріоритеті вільних пар для днів, в яких у викладача є менше 2 навчальних пар занять (@@k < 2):

CREATE proc DoPriorTeachWeek_2
@@id_teach smallint,
@@dow tinyint,
@@Pmin tinyint,
@@Pmax tinyint,
@@k tinyint

Параметри процедури:

@@id_teach – ідентифікатор викладача;

@@dow – номер дня тижня;

@@Pmin – номер першої зайнятої пари протягом дня;

@@Pmax – номер останньої зайнятої пари протягом дня;

@@k – кількість наявних пар занять протягом дня.

Процедура містить розгалужений умовний оператор, який призначає вільним навчальним парам пріоритет для постановки на них заняття у розклад викладача (розділ 6).

Процедури DoPriorTeachWeek2 та DoPriorTeachWeek3 виконують ті ж функції, що і процедура DoPriorTeachWeek_2, але для випадків, коли у викладача є дві пара занять на день (@@k = 2) та більше двох пар.

Для розрахунку пріоритетів вільних пар викладача при постановці у розклад заняття через тиждень використовуються також три процедури: DoPriorTeachWeek_02, DoPriorTeachWeek02, DoPriorTeachWeek03.

Для розрахунку пріоритетів вільних пар груп використовуються процедури з аналогічним програмним кодом, у яких вибірки виконуться не

для викладача з ідентифікатором id_teach = @id_teach, а для групи з ідентифікатором id_gr = @id_gr.

Процедура StudyToTTEveryWeek ставить у розклад щотижневе заняття:

```
CREATE  proc StudyToTTEveryWeek
@id_study int,
@id_gr smallint,
@id_teach smallint,
@id_pare smallint,
@id_aud smallint
```

Параметри процедури:

@id_study – ідентифікатор заняття;

@id_gr – ідентифікатор академічної групи;

@id_teach – ідентифікатор викладача;

@id_pare – ідентифікатор навчальної пари;

@id_aud – ідентифікатор аудиторії.

Процедура StudyToTTEveryWeek додає у таблицю STT (запис з атрибутами (@id_study, @id_pare, @id_aud, NULL).

Для кортежу відношення Study з атрибутом id_study = @id_study ставиться відмітка, що заняття поставлене у розклад: is_tt = 1.

З таблиці вільних пар групи видаляються кортежі зі значенням атрибутів: id_gr = @id_gr, id_pare = @id_pare.

З таблиці вільних пар викладача видаляються кортежі зі значенням атрибутів: id_teach = @id_teach, id_pare = @id_pare.

З таблиці вільних пар аудиторій видаляються кортежі зі значенням атрибутів: id_aud = @id_aud, id_pare = @id_pare.

Процедура StudyToTTWeek ставить у розклад заняття через тиждень:

```
CREATE  proc StudyToTTWeek
@id_study int,
@id_gr smallint,
@id_teach smallint,
@id_pare smallint,
@id_aud smallint,
@weeks bit
```

Параметри процедури:

@weeks – позначка, на який тиждень ставиться заняття у розклад: на парний (@weeks = 1), чи непарний (@weeks = 0).

Інші параметр такі ж, як у процедури StudyToTTEveryWeek.

Процедура StudyToTTWeek додає у таблицю STT (запис з атрибутами (@id_study, @id_pare, @id_aud, @weeks).

Для кортежу відношення Study з атрибутом id_study = @id_study ставиться відмітка, що заняття поставлене у розклад: is_tt = 1.

З таблиці вільних пар групи видаляється кортеж зі значенням атрибутів: id_gr = @id_gr, id_pare = @id_pare, weeks = @weeks. Якщо в таблиці є кортеж з тими ж атрибутами, але weeks = NULL, для цього кортежу значення атрибуту weeks змінюється на протилежне від @weeks.

З таблиці вільних пар викладача видаляються кортежі зі значенням атрибутів: id_teach = @id_teach, id_pare = @id_pare, weeks = @weeks. Якщо в таблиці є кортеж з тими ж атрибутами, але weeks = NULL, для цього кортежу значення атрибуту weeks змінюється на протилежне від @weeks.

З таблиці вільних пар аудиторій видаляються кортежі зі значенням атрибутів: id_aud = @id_aud, id_pare = @id_pare, weeks = @weeks. Якщо в таблиці є кортеж з тими ж атрибутами, але weeks = NULL, для цього кортежу значення атрибуту weeks змінюється на протилежне від @weeks.

9.2 Представлення БД «Розклад занять»

У збережених процедурах БД використовується багато представлень, які є проміжними вибірками для отримання необхідних складних вибірок з таблиць БД «Розклад занять». Але також потрібні представлення, які реалізують вимоги користувачів до зручної роботи з базою даних.

Такими представленнями, наприклад, є представлення, які дозволяють отримати розклад занять заданого викладача або заданої академічної групи у зручному для перегляду вигляді.

Список всіх занять у розкладі для викладача з ідентифікатором id_t = k знаходиться вибіркою з таблиць Study та STT:

$$T\_t\ (id\_st,id\_g,id\_sub,id\_kind,id\_p,id\_a) \leftarrow (\sigma_{id\_t=k}(Study)) \rhd\lhd STT \quad (9.1)$$

Для отримання зручного для користувача виду розкладу занять потрібне природне з’єднання відношення T_t з таблицями-довідниками, що містять дані про назви груп, навчальних дисциплін, видів занять, навчальних пар, аудиторій. При цьому бажано, щоб вся вказана інформація виводилась у два стовпчики:

1) номер пари протягом тижня, наприклад, «Пн-1», «Ср-3»;

2) дані про заняття на цій парі у вигляді рядка тексту.

Вибірку занять розкладу слід доповнити порожніми парами, коли у викладача немає занять.

Оскільки розробляється розклад занять з урахуванням занять через тиждень, то для зручного перегляду розкладу занять потрібно вивести розклад занять на різних тижнях у окремі стовпчики.

Приклад розкладу занять заданого викладача наведено на рис. 9.1.

Пара	нечетная неделя	четная неделя
Пн-1		
Пн-2		
Пн-3		
Пн-4		
Пн-5		
Вт-1		ВишМат_ММДО семинар,310(1) К-21
Вт-2		
Вт-3		
Вт-4	ВишМат_ММДО лекция,409(1) К-21-22	ВишМат_ММДО семинар,310(1) К-22
Вт-5		
Ср-1	ВишМат_ММДО лекция,409(1) К-21-22	ВишМат_ММДО лекция,409(1) К-21-22
Ср-2		
Ср-3		
Ср-4		
Ср-5		
Чт-1		
Чт-2		
Чт-3		
Чт-4		
Чт-5		
Пт-1		
Пт-2		
Пт-3		
Пт-4		
Пт-5		

Рисунок 9.1 – Вигляд розкладу занять викладача

Серверне програмне забезпечення дозволяє складати розклад занять при наявності вихідних даних для нього. Для введення даних та перегляду готового розкладу потрібен набір клієнтських додатків, які можуть бути як офісними додатками, так і веб-додатками.

ВИСНОВКИ

Зручний для студентів та викладачів розклад занять є необхідною складовою ефективного навчального процесу у ЗВО. Розробка програми автоматизованого складання розкладу занять є актуальним завданням, оскільки не існує подібних програм з відкритим кодом.

Задачею комплексної магістерської роботи була розробка серверного програмного забезпечення для програми автоматизованого складання розкладу занять у ЗВО. Задачею даної частини комплексної магістерської роботи було розробка моделі та алгоритму урахуванні у програмі занять з різною кількістю годин на тиждень.

Результатом виконання комплексної магістерської роботи є розробка набору збережених процедур та представлень БД, які дозволяють автоматизовано складати розклад занять у ЗВО.

Для надійної роботи програми бажано складати розклад занять у декілька кроків, задаючи на кожному кроці факультет та курс, для груп яких буде складатись розклад занять.

Процедура складання розкладу занять містить декілька емпіричних констант, можливі значення для яких наводяться у даній частині роботи. При впровадженні програми бажано провести дослідження для визначення оптимальних значень всіх емпіричних констант.

Результатами виконання даної частини магістерської роботи є:

- розроблена модель обліку занять з різною кількістю годин на тиждень в програмі складання розкладу занять;
- відповідно до моделі реструктурована база даних;
- розроблений алгоритм обліку занять з різною кількістю годин на тиждень в програмі складання розкладу занять;
- розроблений алгоритм сортування списку занять для постановки в розклад з обліком занять, які проводяться через тиждень;
- розроблено серверне програмне забезпечення програми автоматизованого складання розкладу занять;
- Розроблене серверне програмне забезпечення включає в себе 11 збережених процедур і 6 представлень на мові T-SQL для СУБД MS SQL Server 2005

Програма автоматизованого складання розкладу занять розробляється як частина інформаційної системи «Навчальний процес університету», тому не вимагає введення великих масивів вихідних даних.

ПЕРЕЛІК ПОСИЛАННЬ

1 Куликов Г. Г., Никулина Н. О., Речкалов А. В. Управление проектами на основе системного моделирования: Учебное пособие. Уфа: УГАТУ, 2009. − 171 с.

2 Клиффорд Ф. Грей, Эри У. Ларсон. Управление проектами. М: ДиС, 2007. − 608 с.

3 Мазур И. И., Шапиро В. Д, Ольдерогге Н. Г. Управление проектами: учеб. пособие. М.: Омега-Л, 2005. – 194 с.

4 Вендров А. М. CASE-технологии. Современные методы и средства проектирования информационных систем. М: «Финансы и статистика», 1998. − 98 с.

5 Новиков Ф.А. Дискретная математика для программистов. – СПб: Питер, 2000. – 304 с.

6 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.

7 Дискретная математика. Конспект лекций. 3. Алгебра отношений. URL: http://docplayer.ru/47041850-Diskretnaya-matematika-konspekt-lekciy-3-algebra-otnosheniy.html (дата обращения: 11.01.2018).

8 Костюк В.И., Мартинес Х.О., Зорин В.В. Использование алгоритмов последовательной обработки для составления расписаний / Вопросы создания АСУ ВУЗ. М.: НИИВШ, 1976. – С.3-5.

9 Бартенев А.С. Обзор основных вопросов автоматизированного составления расписания занятий в высшем учебном заведении. // Современные научные исследования и инновации. Сентябрь, 2011. URL: http://web.snauka.ru/issues/2011/09/2576 (дата обращения: 11.01.2018).

10 Лупин С.А., Милехина Т.В. Статья «Метод решения задач составления расписания, ориентированный на кластерные вычислительные системы», Научно-технический журнал “Известия высших учебных заведений. Электроника”, МИЭТ №6 2007 г., С. 63 – 70.

11 Верёвкин В.И., Исмагилова О.М., Атавин Т.А. Автоматизированное составление расписания учебных занятий вуза с учётом трудности дисциплин и утомляемости студентов. Доклады ТУСУРа, №1 (19). часть 1, 2009. С.221-225.

12 Береговых Ю.В., Васильев Б.А., Володин Н.А. Алгоритм составление расписания занятий / Искусственный интеллект. – 2009. – №2 – С.50-56

13 Кузнецов А.А. Разработка автоматизированной системы составления расписания для педагогического Вуза // Современные научные исследования и инновации. 2016. № 9 [Электронный ресурс]. URL: http://web.snauka.ru/issues/2016/09/67427 (дата обращения: 11.01.2018).

14 Галузин К.С. Разработка модуля для автоматизации составления оптимального учебного расписания в рамках единой информационной системы образовательного учреждения / К.С. Галузин, Столбов В.Ю. // Известия Белорусской инженерной академии. – 2003. – 43-50 с.

15 Дейт К. Дж.. Введение в системы баз данных, 6-е издание: Пер. с англ. К.; М.; СПб: Издательский дом «Вильямс», 2000 – 848 с.

16 Введение в MS SQL Server и T-SQL [Электронный ресурс]. URL: https://metanit.com/sql/sqlserver/1.1.php (дата обращения: 11.01.2018).

17 Мамаев Е.В. Microsoft SQL Server 2000. – СПб.: БХВ-Петербург, 2004. – 1260 с.

18 Сравнение SQL баз данных. [Электронный ресурс]. URL: http://usanov.net/1970-sravnenie-sql-baz-dannyx (дата обращения: 11.01.2018).

19 Дэвидсон Л. Проектирование баз данных на SQL Server 2000 / Л.Дэвидсон; пер. с англ. М. БИНОМ. Лаборатория знаний, 2003. – 680 с.

20 Роберт Виейра. Программирование баз данных Microsoft SQL Server 2005. Базовый курс. М.: «Диалектика», 2007. – 832 с.

Д О Д А Т К И

ДОДАТОК А

Логічна схема БД «Розклад занять» – підсхема «Вихідні дані»

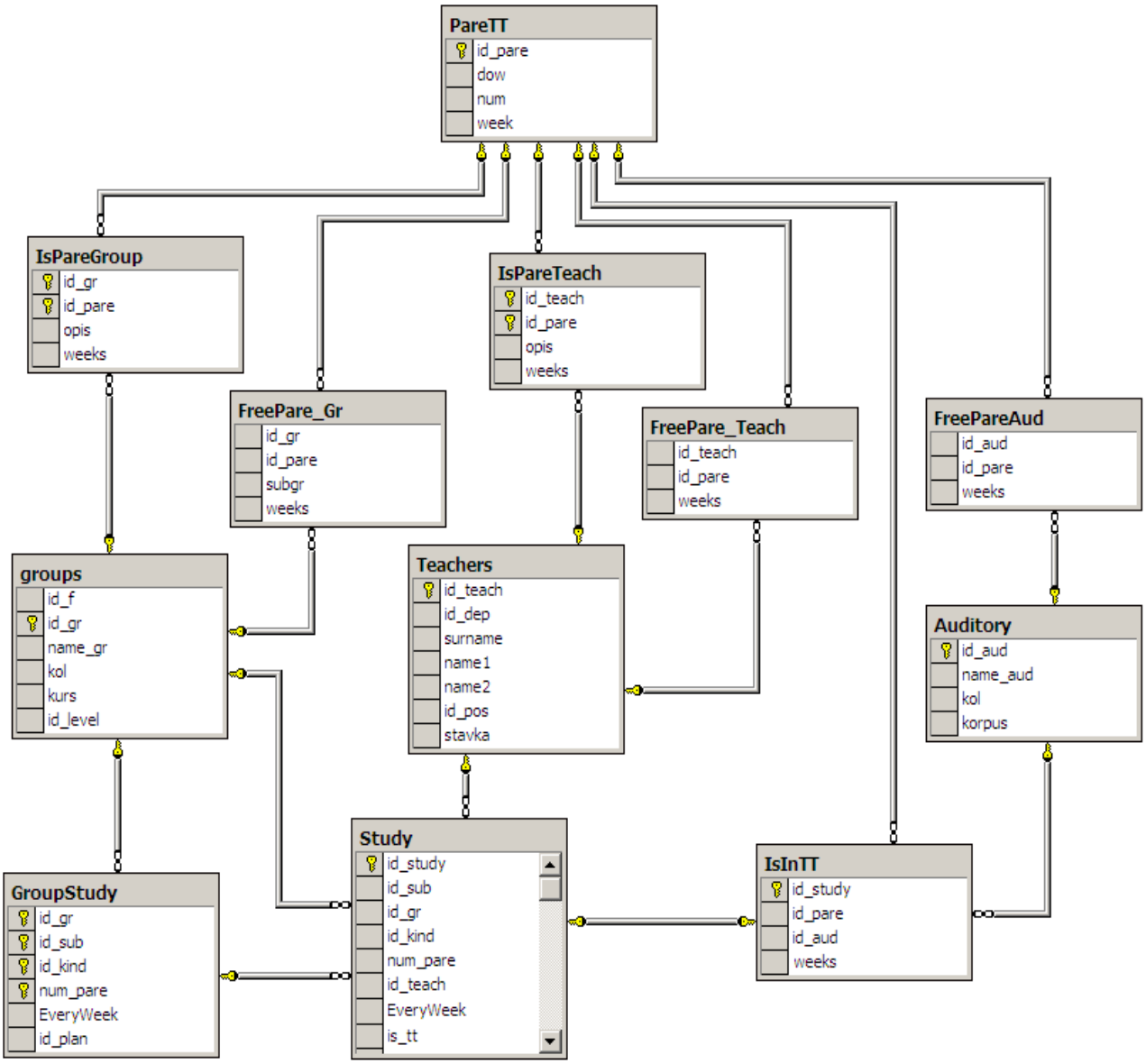

Рисунок А.1 – Логічна схема БД «Розклад занять» – підсхема «Вихідні дані»

ДОДАТОК Б

Вихідний код серверного ПЗ

```
CREATE proc MakeTimeTable
@k1 tinyint,
@k2 tinyint,
@kp_st tinyint,
@TeachGr numeric(3,2),
@nMax int,
@id_st int output,
@k tinyint output,
@msg varchar(200) output
as
BEGIN
      declare @n1 int, @nt int, @id_gr smallint, @id_study int, @@i int
      DECLARE @@id_gr smallint,@@id_teach smallint, @@id_fl smallint, @@n
smallint, @@k1 smallint
      Declare @@id_study int, @@id_pare smallint, @@id_g smallint, @@prior
int, @@m int
      DECLARE @@dow tinyint,@@Pmin tinyint,@@Pmax tinyint, @@k tinyint,
@@id_aud smallint
      DECLARE @@subgr bit, @@week bit, @@weeks bit, @@st bit
      DECLARE @@id_sub smallint, @@num_pare tinyint, @@id_kind tinyint

      set @@st=0
      set @n1=(select k=count(*) from Study where is_tt=0)
      if @n1=0
      begin
            set @k=1
            set @nt=(select k=count(*) from Study where is_tt is null)
            set @n1=(select k=count(*) from Study where is_tt=1)
            set @msg='Нет списка занятий для постановки в расписание. Есть
'+str(@n1,4)+' занятий в расписании и '+
      str(@nt,4)+' занятий, которые не должны ставиться в расписание'
      end
      else
      begin
         set @@i = 0
         while @n1>0 and @@i<@nMax
         begin
            delete StudyFull

            insert into StudyFull
            select * from FillStudyFullLect
            insert into StudyFull
            select * from FillStudyFullPract
            insert into StudyFull
            select * from FillStudyFullLab

            delete StudyFull where EveryWeek=0 and weeks is null

            set @nt=(select n=count(*) from ViewTeachDensity where
kp_st<@kp_st and id_teach in(select id_teach from Study where is_tt=0))
            if @nt>0
            begin
                  set @@k1=(select k=min(K1) from ViewStudyK1K2)
                  if @@k1>1
                        set @@st=1
            end
            if @@st=1
```

```
            begin
                  DECLARE SortList_CURSOR CURSOR FAST_FORWARD FOR
                  SELECT s.id_study,s.id_gr,s.id_teach,s.id_fl, s.subgr,
s.EveryWeek,tt.k_1
                  FROM Study s join
                  (select TOP 1 k.*,t.*, k_1 = case when K1>@k1 then @k1 else
K1 end,
                              k_2 = case when K2>@k2 then @k2 else K2
end,k_gr=s.CountGr,
                              k_pst=case when kp_st>@kp_st then @kp_st else
kp_st end
                        from  ViewStudyK1K2 k join study s on
k.id_study=s.id_study
                              join ViewTeachDensity t on s.id_teach=t.id_teach
                         order by k_pst, kstp desc, k_1, k_2, k_gr desc) tt on
s.id_study=tt.id_study
                  OPEN SortList_CURSOR
                  FETCH NEXT FROM SortList_CURSOR INTO
@@id_study,@@id_gr,@@id_teach, @@id_fl, @@subgr, @@week,@@k1

                  CLOSE SortList_CURSOR
                  DEALLOCATE SortList_CURSOR
            end
            else
            begin
            --нет преподавателей с очень плотной нагрузкой

                  DECLARE SortList_CURSOR CURSOR FAST_FORWARD FOR
                  SELECT s.id_study,s.id_gr,s.id_teach,s.id_fl, s.subgr,
s.EveryWeek,tt.k_1
                  FROM Study s join
                  (select TOP 1 k.*,t.*, k_1 = case when K1>@k1 then @k1 else
K1 end,
                        k_2 = case when K2>@k2 then @k2 else K2
end,k_gr=s.CountGr,
                        k_pst=case when kp_st>@kp_st then @kp_st else kp_st
end
                     from  ViewStudyK1K2 k join study s on
k.id_study=s.id_study
                              join ViewTeachDensity t on s.id_teach=t.id_teach
                        order by k_1, k_2, k_gr desc,k_pst, kstp desc) tt on
s.id_study=tt.id_study
                  OPEN SortList_CURSOR
                  FETCH NEXT FROM SortList_CURSOR INTO
@@id_study,@@id_gr,@@id_teach, @@id_fl, @@subgr, @@week,@@k1

                  CLOSE SortList_CURSOR
                  DEALLOCATE SortList_CURSOR
            end

            if @@k1>0
            begin
                  if @@k1=1
                  begin
                        DECLARE PareAud_CURSOR CURSOR FAST_FORWARD FOR
                        SELECT TOP 1 id_pare, id_aud,weeks from StudyFull s
                        where id_study=@@id_study order by s.prioritet
                        OPEN PareAud_CURSOR
                        FETCH NEXT FROM PareAud_CURSOR INTO @@id_pare,
@@id_aud, @@weeks
                        CLOSE PareAud_CURSOR
                        DEALLOCATE PareAud_CURSOR
                  end
                  else
```

```
                  begin
                  exec FillTeachParePriorWeek @@id_teach, @@id_study, @@week

                        if @@id_fl=0 or @@id_fl is null
                        begin
                              exec FillGrParePriorWeek @@id_gr, @@id_study,
@@week
                              if @@week =1
                              begin
                                    DECLARE PareAud_CURSOR CURSOR FAST_FORWARD
FOR
                                    SELECT TOP 1 s.id_pare, s.id_aud
                                    from StudyFull s join GrParePrior g on
s.id_gr=g.id_gr and s.id_pare=g.id_pare join
                                          TeachParePrior t on
s.id_teach=t.id_teach and s.id_pare=t.id_pare
                                    where id_study=@@id_study
                                    order by g.prioritet+@TeachGr*t.prioritet
desc, s.prioritet
                                    OPEN PareAud_CURSOR
                                    FETCH NEXT FROM PareAud_CURSOR INTO
@@id_pare, @@id_aud
                                    CLOSE PareAud_CURSOR
                                    DEALLOCATE PareAud_CURSOR
                              end
                              else
                              begin
                                    DECLARE PareAud_CURSOR CURSOR FAST_FORWARD
FOR
                                    SELECT TOP 1 s.id_pare, s.id_aud, s.weeks
                                    from StudyFull s join GrParePriorWeek g on
s.id_gr=g.id_gr and s.id_pare=g.id_pare and s.weeks=g.weeks
                                          join TeachParePriorWeek t on
s.id_teach=t.id_teach and s.id_pare=t.id_pare and s.weeks=t.weeks
                                    where id_study=@@id_study and s.weeks is
not null and g.weeks is not null and t.weeks is not null
                                    order by g.prioritet+@TeachGr*t.prioritet
desc, s.prioritet
                                    OPEN PareAud_CURSOR
                                    FETCH NEXT FROM PareAud_CURSOR INTO
@@id_pare, @@id_aud,@@weeks
                                    CLOSE PareAud_CURSOR
                                    DEALLOCATE PareAud_CURSOR
                              end
                        end
                        else
                        begin
                              exec FillFlowParePriorWeek  @@id_study,@@id_fl,
@@week

                              if @@week =1
                              begin
                                    DECLARE PareAud_CURSOR CURSOR FAST_FORWARD
FOR
                        SELECT TOP 1 s.id_pare, s.id_aud
                        from StudyFull s join FlowPrior g on
s.id_pare=g.id_pare join
                        TeachParePrior t on s.id_teach=t.id_teach and
s.id_pare=t.id_pare
                        where id_study=@@id_study
                        order by g.Psum+@TeachGr*t.prioritet desc, s.prioritet
                        OPEN PareAud_CURSOR
      FETCH NEXT FROM PareAud_CURSOR INTO @@id_pare, @@id_aud
                        CLOSE PareAud_CURSOR
```

```
                              DEALLOCATE PareAud_CURSOR
                              end
                              else
                              begin
                              DECLARE PareAud_CURSOR CURSOR FAST_FORWARD FOR
                              SELECT TOP 1 s.id_pare, s.id_aud, s.weeks
                              from StudyFull s join FlowPrior g on
s.id_pare=g.id_pare and s.weeks=g.weeks
                              join TeachParePriorWeek t on
s.id_teach=t.id_teach and s.id_pare=t.id_pare and s.weeks=t.weeks
                              where id_study=@@id_study and s.weeks is not
null and g.weeks is not null and t.weeks is not null
                              order by g.Psum+@TeachGr*t.prioritet desc,
s.prioritet
                              OPEN PareAud_CURSOR
                              FETCH NEXT FROM PareAud_CURSOR INTO @@id_pare,
@@id_aud,@@weeks
                              CLOSE PareAud_CURSOR
                              DEALLOCATE PareAud_CURSOR
                              end
                        end
                  end

                  if @@week =1
                  begin
                        if @@subgr is null
                              exec StudyToTTEveryWeek
@@id_study,@@id_gr,@@id_teach,@@id_fl ,@@id_pare,@@id_aud
                        else
                              exec StudyToTTSubgr
@@id_study,@@id_gr,@@id_teach,@@id_pare,@@id_aud,@@subgr
                  end
                  else
                  begin
                        if @@subgr is null
                              exec StudyToTTWeek
@@id_study,@@id_gr,@@id_teach,@@id_fl ,@@id_pare,@@id_aud,@@weeks
                        else
                              exec StudyToTTWeekSubgr
@@id_study,@@id_gr,@@id_teach,@@id_pare,@@id_aud,@@weeks ,@@subgr
                  end
            end
            else
            begin
                  set @id_st=@@id_study
                  set @k=2
                  set @msg ='тупиковая ситуация для занятия
№'+convert(char(5),@id_st)
                  return
            end
            set @n1=@n1 - 1
            set @@i=@@i + 1
         end
        if @n1=0
        begin
            set @k=0
            set @msg='Расписание составлено'
        end
        else
        if @@i=@nMax
        begin
            set @k=0
            set @msg='В расписание поставлено '+str(@nMax,5)+' занятий'
        end
```

```
      end
END
GO


CREATE   proc FillTeachParePriorWeek
@id_teach smallint,
@id_study int,
@EveryWeek bit
as
BEGIN
      DECLARE @@dow tinyint,@@Pmin tinyint,@@Pmax tinyint, @@k tinyint

      delete ParePriorTeachWeek where id_teach=@id_teach
      delete TeachParePriorWeek where id_teach=@id_teach
      delete TeachParePrior where id_teach=@id_teach
      delete ParePriorTeach where id_teach=@id_teach

      insert into TeachParePriorWeek
      select id_pare, is_st=0, id_teach=@id_teach, prioritet=null,weeks=0
      from NotPareTeach where id_teach=@id_teach

      insert into TeachParePriorWeek
      select id_pare, is_st=0, id_teach=@id_teach, prioritet=null,weeks=1
      from NotPareTeach where id_teach=@id_teach

      insert into TeachParePrior
      select id_pare, is_st=0, id_teach=@id_teach, prioritet=null
      from NotPareTeach where id_teach=@id_teach

      insert into TeachParePriorWeek
      select id_pare, is_st=1, id_teach=@id_teach, prioritet=null, weeks=0
      from (select id_pare from  IsPareTeach where id_teach=@id_teach and
weeks is null or weeks=0
            union
            select id_pare from Study s join IsInTT t on s.id_study=t.id_study
            where id_teach=@id_teach and OddWeek is null or OddWeek=0) tt

      insert into TeachParePriorWeek
      select id_pare, is_st=1, id_teach=@id_teach, prioritet=null, weeks=1
      from (select id_pare from  IsPareTeach where id_teach=@id_teach and
weeks is null or weeks=1
            union
            select id_pare from Study s join IsInTT t on s.id_study=t.id_study
            where id_teach=@id_teach and OddWeek is null or OddWeek=0) tt

      insert into TeachParePrior
      select id_pare, is_st=1, id_teach=@id_teach, prioritet=null
      from (select id_pare from  IsPareTeach where id_teach=@id_teach
            union
            select id_pare from Study s join IsInTT t on s.id_study=t.id_study
            where id_teach=@id_teach) tt

      insert into TeachParePriorWeek
      select id_pare, is_st=0, id_teach=@id_teach, prioritet=0, weeks=0
      from PareTT
      where id_pare not in (select id_pare from TeachParePriorWeek where
id_teach=@id_teach and weeks=0)

      insert into TeachParePriorWeek
      select id_pare, is_st=0, id_teach=@id_teach, prioritet=0, weeks=1
      from PareTT
      where id_pare not in (select id_pare from TeachParePriorWeek where
id_teach=@id_teach and weeks=1)
```

```
      insert into TeachParePrior
      select id_pare, is_st=0, id_teach=@id_teach, prioritet=0
      from PareTT
      where id_pare not in (select id_pare from TeachParePrior where
id_teach=@id_teach)

      update TeachParePrior set prioritet=null
      where id_teach=@id_teach and id_pare not in
      (select id_pare from StudyFull where id_study =@id_study)

      update TeachParePriorWeek set prioritet=null
      where id_teach=@id_teach and id_pare not in
      (select id_pare from MustPareTeach where id_teach=@id_teach)

      update TeachParePrior set prioritet=null
      where id_teach=@id_teach and id_pare not in
      (select id_pare from MustPareTeach where id_teach=@id_teach)

      if @EveryWeek=0
      begin
            update TeachParePriorWeek set prioritet=null
            where weeks=0 and id_teach=@id_teach and id_pare not in
            (select id_pare from IsInStudyFullTeach where id_study =@id_study
and weeks=0)

            update TeachParePriorWeek set prioritet=null
            where weeks=1 and id_teach=@id_teach and id_pare not in
            (select id_pare from IsInStudyFullTeach where id_study =@id_study
and weeks=1)
      end

      DECLARE WorkPareTeach_CURSOR CURSOR FAST_FORWARD FOR
      select dow, Pmin=min(num), Pmax=max(num), k=count(*)
      from
        (select t.*, dow,num from PareTT p join TeachParePrior t on
t.id_pare=p.id_pare where id_teach=@id_teach) tt
      where is_st=1
      group by all dow
      OPEN WorkPareTeach_CURSOR
      FETCH NEXT FROM WorkPareTeach_CURSOR INTO @@dow,@@Pmin,@@Pmax, @@k
      WHILE @@FETCH_STATUS = 0
      BEGIN
            if @EveryWeek=1
            begin
                  if @@k<2
                  exec DoPriorTeachWeek_2
@id_teach ,@@dow ,@@Pmin ,@@Pmax ,@@k
                  else
                  begin
                  if @@k=2
                  exec DoPriorTeachWeek2 @id_teach ,@@dow ,@@Pmin ,@@Pmax
                  else
                  exec DoPriorTeachWeek3 @id_teach ,@@dow ,@@Pmin ,@@Pmax ,@@k
                  end
            end
            else
            begin
            if @@k<2
            exec DoPriorTeachWeek0_2 @id_teach ,@@dow ,@@Pmin ,@@Pmax ,@@k
            else
            if @@k=2
                  exec DoPriorTeachWeek02 @id_teach ,@@dow ,@@Pmin ,@@Pmax
                  else
```

```
            exec DoPriorTeachWeek03 @id_teach ,@@dow ,@@Pmin ,@@Pmax ,@@k
            end
            FETCH NEXT FROM WorkPareTeach_CURSOR INTO @@dow,@@Pmin,@@Pmax, @@k
      END
      CLOSE WorkPareTeach_CURSOR
      DEALLOCATE WorkPareTeach_CURSOR

      if @EveryWeek=0
      update TeachParePriorWeek set prioritet=p.prioritet
      from TeachParePriorWeek t join ParePriorTeachWeek p on
t.id_teach=p.id_teach and t.id_pare=p.id_pare and t.weeks=p.weeks
      where t.is_st=0 and t.prioritet is not null and  p.weeks is not null and
t.id_teach=@id_teach

      if @EveryWeek=1
      update TeachParePrior set prioritet=p.prioritet
      from TeachParePrior t join ParePriorTeachWeek p on t.id_teach=p.id_teach
and t.id_pare=p.id_pare
      where t.is_st=0 and t.prioritet is not null and p.weeks is null and
t.id_teach=@id_teach
END
GO


CREATE  proc StudyToTTEveryWeek
@id_study int,
@id_gr smallint,
@id_teach smallint,
@id_pare smallint,
@id_aud smallint
as
BEGIN
      delete FreePare_Teach where id_teach=@id_teach and id_pare=@id_pare
      delete FreePareAud where id_aud=@id_aud and id_pare=@id_pare

      delete StudyFull where id_aud=@id_aud and id_pare=@id_pare
      delete StudyFull where id_teach=@id_teach and id_pare=@id_pare

      update Study set is_tt=1 where id_study=@id_study
      insert IsInTt (id_study,id_pare,id_aud, OddWeek)
      values(@id_study,@id_pare,@id_aud, null)
      delete StudyFull where id_study=@id_study

      delete FreePare_Gr where id_gr=@id_gr and id_pare=@id_pare
      delete StudyFull where id_gr=@id_gr and id_pare=@id_pare

END
GO

CREATE  proc StudyToTTWeek
@id_study int,
@id_gr smallint,
@id_teach smallint,
@id_pare smallint,
@id_aud smallint,
@weeks bit
as
BEGIN
      DECLARE @@weeks bit, @@n tinyint, @@subgr bit, @@k tinyint

      delete FreePare_Teach where id_teach=@id_teach and id_pare=@id_pare and
weeks=@weeks
      update FreePare_Teach set weeks=1-@weeks where id_teach=@id_teach and
id_pare=@id_pare and weeks is null
```

```
      delete FreePareAud where id_aud=@id_aud and id_pare=@id_pare and
weeks=@weeks
      update FreePareAud  set weeks=1-@weeks where id_aud=@id_aud and
id_pare=@id_pare and weeks is null

      delete StudyFull where id_aud=@id_aud and id_pare=@id_pare and
(weeks=@weeks or weeks is null)
      delete StudyFull where id_teach=@id_teach and id_pare=@id_pare and
(weeks=@weeks or weeks is null)

      update Study set is_tt=1 where id_study=@id_study
      insert IsInTt (id_study,id_pare,id_aud, OddWeek)
values(@id_study,@id_pare,@id_aud, @weeks)

      delete StudyFull where id_study=@id_study
      delete FreePare_Gr where id_gr=@id_gr and id_pare=@id_pare and
weeks=@weeks
      update FreePare_Gr set weeks=1-@weeks where id_gr=@id_gr and
id_pare=@id_pare and weeks is null
      delete StudyFull where id_gr=@id_gr and id_pare=@id_pare and
(weeks=@weeks or weeks is null)

END
GO


CREATE  proc DoPriorGroupWeek_2
@@id_gr smallint,
@@dow tinyint,
@@Pmin tinyint,
@@Pmax tinyint,
@@k tinyint
AS
BEGIN
    DECLARE @p1 int, @p2 int, @p3 int, @p4 int, @p5 int ,@p6 int, @p7 int,
@p8 int, @p9 int
    DECLARE @p10 int, @p11 int, @p12 int, @p13 int ,@p14 int, @p15 int, @p16
int, @p25 int
    DECLARE @p17 int, @p18 int, @p19 int, @p20 int, @p21 int ,@p22 int, @p23
int, @p24 int
    DECLARE @week1 bit,@week2 bit, @st1 bit, @st2 bit,  @num tinyint
      set @p25=1
      set @p24=2
      set @p23=3
      set @p22=4
      set @p21=5
      set @p20=6
      set @p19=7
      set @p18=8
      set @p17=9
      set @p16=10
      set @p15=11
      set @p14=12
      set @p13=13
      set @p12=14
      set @p11=16
      set @p10=17
      set @p9 =19
      set @p8 =20
      set @p7=21
      set @p6=22
      set @p5=24
      set @p4=25
```

```
      set @p3=28
      set @p2=29
      set @p1=32

      if @@k=0
      BEGIN
            insert into ParePriorGrWeek
            select id_gr=@@id_gr , id_pare, weeks=null, prioritet=@p15
            from pareTT where dow=@@dow and num in (1,2)

            insert into ParePriorGrWeek
            select id_gr=@@id_gr , id_pare, weeks=null, prioritet=@p18
            from pareTT where dow=@@dow and num =3

            insert into ParePriorGrWeek
            select id_gr=@@id_gr , id_pare, weeks=null, prioritet=@p20
            from pareTT where dow=@@dow and num =4

            insert into ParePriorGrWeek
            select id_gr=@@id_gr , id_pare, weeks=null, prioritet=@p25
            from pareTT where dow=@@dow and num =5

      END
      ELSE
      BEGIN
            set @st1 =(select is_st from GrParePriorWeek where id_gr=@@id_gr
and weeks=0 and id_pare=@@dow*10+@@Pmin)
            set @st2 =(select is_st from GrParePriorWeek where id_gr=@@id_gr
and weeks=1 and id_pare=@@dow*10+@@Pmin)
            if @st1=1
            begin
                  if @st2 =1
                  set @week1=null
                  else
                  set  @week1=0
            end
            else
            set  @week1=1

            if @week1 is null
            begin
                  if @@Pmin>1
                  BEGIN
                        Insert into ParePriorGrWeek
                        Select id_gr=@@id_gr ,id_pare,weeks=null,
prioritet=@p9
                        From pareTT Where dow=@@dow and num =@@Pmin-1

                        if @@Pmin<4
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p10
                              From pareTT Where dow=@@dow and num =@@Pmin+1
                        if @@Pmin>2
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p19
                              From pareTT Where dow=@@dow and num =@@Pmin-2
                        if @@Pmin=2
                        begin
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p23
                              From pareTT Where dow=@@dow and num =4
```

```
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p25
                              From pareTT Where dow=@@dow and num = 5
                        end
                        if @@Pmin=3
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p25
                              From pareTT Where dow=@@dow and num = 5
                        if @@Pmin=4
                        begin
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p21
                              From pareTT Where dow=@@dow and num = 1

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p25
                              From pareTT Where dow=@@dow and num =5
                        end
                        if @@Pmin=5
                        begin
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p25
                              From pareTT Where dow=@@dow and num = 2

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=0
                              From pareTT Where dow=@@dow and num = 1
                        end
                  END
                  ELSE
                  BEGIN
                        Insert into ParePriorGrWeek
                        Select id_gr=@@id_gr ,id_pare,weeks=null,
prioritet=@p9
                        From pareTT Where dow=@@dow and num = 2

                        Insert into ParePriorGrWeek
                        Select id_gr=@@id_gr ,id_pare,weeks=null,
prioritet=@p15
                        From pareTT Where dow=@@dow and num = 3

                        Insert into ParePriorGrWeek
                        Select id_gr=@@id_gr ,id_pare,
weeks=null,prioritet=@p24
                        From pareTT Where dow=@@dow and num =4

                        Insert into ParePriorGrWeek
                        Select id_gr=@@id_gr ,id_pare,weeks=null, prioritet=0
                        From pareTT Where dow=@@dow and num = 5
                  END
            end
            else
            begin
                  if @@Pmin<3
                  BEGIN
                        Insert into ParePriorGrWeek
```

```
                        Select id_gr=@@id_gr ,id_pare,weeks=null,
prioritet=@p8
                        From pareTT Where dow=@@dow and num =@@Pmin+1

                        if @@Pmin=1
                        BEGIN
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p10
                              From pareTT Where dow=@@dow and num =3

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p16
                              From pareTT Where dow=@@dow and num =4

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=0
                              From pareTT Where dow=@@dow and num =5
                        END
                        if @@Pmin=2
                        begin
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p14
                              From pareTT Where dow=@@dow and num =1

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p12
                              From pareTT Where dow=@@dow and num =4

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p25
                              From pareTT Where dow=@@dow and num = 5
                        End
                  END
                  ELSE
                  BEGIN
                        if @@Pmin=3
                        begin
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p10
                              From pareTT Where dow=@@dow and num =1

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p8
                              From pareTT Where dow=@@dow and num =2

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p16
                              From pareTT Where dow=@@dow and num = 4

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p25
                              From pareTT Where dow=@@dow and num = 5
                        End
                        if @@Pmin=4
```

```
                        begin
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p10
                              From pareTT Where dow=@@dow and num =2

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p8
                              From pareTT Where dow=@@dow and num =3

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p18
                              From pareTT Where dow=@@dow and num = 1

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p25
                              From pareTT Where dow=@@dow and num =5
                        end
                        if @@Pmin=5
                        begin
                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p10
                              From pareTT Where dow=@@dow and num =3

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p8
                              From pareTT Where dow=@@dow and num =4

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=@p24
                              From pareTT Where dow=@@dow and num = 2

                              Insert into ParePriorGrWeek
                              Select id_gr=@@id_gr ,id_pare,weeks=null, prior-
itet=0
                              From pareTT Where dow=@@dow and num = 1
                        end
                  END
            end
      END
      Insert into ParePriorGrWeek
      Select id_gr=@@id_gr ,id_pare,weeks=null, prioritet=0
      From pareTT
      Where dow=@@dow and num >5
END
GO
```