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 груп та потоків різного розміру»

Виконав студент 2 курсу групи МК- 61
спеціальності 122 Комп'ютерні науки та інформаційні технології
Домчинська Аліса Олександрівна

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

Консультант

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

Одеса 2018

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ЗВО – заклад вищої освіти

ІС – інформаційна система

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)

FPF – ім’я відношення, що описує вільні навчальні пари академічного потоку (Free Pare Flow)

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. Існує навіть наука з назвою «Теорія розкладу», проте, вона дозволяє отримати чітке рішення лише для обмеженого круга задач. Задача складання розкладу занять до них не відноситься, вона є складною багатокритеріальною задачею.

Ця задача виникає з року в рік у будь-якій навчальній установі. Деякі заклади вищої освіти використовують різні програми складання розкладу занять, але у багатьох закладах досі складають розклад занять вручну. Обмежене використання автоматизації при складанні розкладу занять у вищі обумовлене відсутністю відкритого алгоритму та програмного коду для розв'язання цієї задачі. Готові програмні рішення коштують дорого і доступні не всім ЗВО. 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.

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.

Отже, хоча 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 потокових лекцій та занять з поділом академічної групи на підгрупи. Стандартним заняттям вважається заняття одного виду (лекція, практичне, лабораторне) з однієї дисципліни впродовж двох академічних годин, тобто вказані види занять є нестандартними і потребують окремої моделі для опису та окремого алгоритму для постановки їх у розклад при автоматизованому складанні розкладу занять.

Дослідження методів вирішення задачі складання розкладу занять при наявності потокових лекцій та занять з поділом академічної групи на підгрупи є завданням цієї магістерської роботи.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

Навчальний процес у закладах вищої освіти має відмінності у порівнянні з навчальним процесом у закладах середньої освіти. Відмінність полягає не тільки в великій кількості навчальних планів, за якими навчаються студенти, але і в більшій різноманітності видів заняття у вишах, наприклад, наявності потокових лекцій. Розробка окремої програми складання розкладу занять в університеті потребує введення великої кількості вихідних даних. Урахування занять з об'єднанням академічних груп у потоки або розділення груп на підгрупи ще збільшує обсяг даних.

Розробка програми у якості складової частини інформаційної системи «Навчальний процес університету» дозволяє використовувати в програмі дані, які отримані на інших стадіях навчального процесу. На початку складання розкладу занять навчальний відділ вже має у базі даних необхідну інформацію для розкладу занять на наступний семестр.

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

1.1 Особливості навчального процесу у закладі вищої освіти

У закладі вищої освіти студент набуває освіту за допомогою аудиторних занять та самостійної роботи. Аудиторні заняття можуть бути трьох видів: лекції, практичні заняття (семінари), лабораторні роботи.

Семінари завжди проводяться для однієї академічної групи. Лекції можуть проводитись для однієї групи, але частіше академічні групи об'єднують у потоки для читання лекцій. Об'єднання груп у потоки для заданої навчальної дисципліни можливе, якщо у навчальних планах всіх груп потоку для цієї дисципліни передбачена однакова кількість годин на тиждень. Кількість годин практичних занять, лабораторних робіт, самостійної підготовки з даної дисципліни у навчальних планах академічних груп одного потоку можуть відрізнятись.

Лабораторні заняття проводяться у спеціалізованих аудиторіях, які звичайно мають менше посадкових місць, ніж аудиторії для проведення семінарських занять. Якщо посадкових місць в спеціалізованій аудиторії не менше, ніж кількість студентів академічної групи, то лабораторні заняття з дисципліни можуть проводитись для всієї групи разом. Якщо спеціалізована аудиторія замала для проведення занять всієї групи, групу розбивають на підгрупи, для яких проводяться окремі заняття. Заняття для обох підгруп одночасно можуть проводитись лише у випадку, коли є принаймні дві

спеціалізованих аудиторії з необхідним обладнанням, та якщо для проведення даних лабораторних занять у групі призначені два викладачі, а не один.

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. При плануванні аудиторного навантаження викладачів враховується об'єднання груп у потоки для читання лекцій та розділення їх на підгрупи для проведення лабораторних робіт

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. Уповноваженими особами кафедри, або на засіданні кафедри вирішується, які дисципліни будуть читатись з обєднанням груп у потоки. Одна академічна група може слухати лекції з різних дисциплін у різних академічних потоках.

Також кафедри визначають, які групи потрібно розділити на підгрупи для проведення лабораторних занять з дисциплін кафедри. При цьому для кожної з дисциплін, які закріплені за однією кафедрою, необхідність розподілення груп на підгрупи для проведення лабораторних робіт розглядається окремо у залежності від розміру спеціалізованої аудиторії та її обладнання. Для вирішення питання, чи потрібно виконувати розділення групи на підгрупи для проведення занять у спеціалізованих аудиторіях кафедри необхідні дані по контингенту студентів на наступний навчальний семестр. Ці дані кафедри отримують з деканатів факультетів разом з навчальними планами академічних груп.

Кафедра надає у навчальну частину ЗВО дані про обєднанням груп у потоки для проведення лекцій з дисциплін кафедр, розділення груп на підгрупи для проведення лабораторних робіт, допустимі аудиторії для проведення кожного заняття, та розподіл занять між викладачами кафедри.

Ці дані є складовою частиною вихідних даних для складання розкладу занять у ЗВО.

Якщо у ЗВО використовується інформаційна система «Навчальний процес університету», то різні суб'єкти навчального процесу отримують готові дані для їх роботи з бази даних; ці дані потрапляють у БД при виконанні іншими користувачами ІС свого циклу робіт. Для відстеження руху даних між суб'єктами та об'єктами навчального процесу потрібно

виконати моделювання інформаційної системи «Навчальний процес університету», зокрема побудувати діаграму потоків даних, що існують в університеті в процесі складання розкладу занять.

1.2 Облік потокових занять та розподілу груп на підгрупи у структурі бази даних

Деканатии факультетів закріплюють за академічними групами навчальні плани на розрахунковий рік. У навчальних планах для кожної дисципліни вказується кількість годин для кожного виду занять: лекцій, семінарів, лабораторних занять. Всі три виду занять по одній дисципліні зустрічаються досить рідко.

З навчального плану, закріпленого за групою на наступний семестр, визначається множина аудиторних занять групи. Якщо всі заняття всіх академічних груп проводяться для однієї цілої групи, то кількість занять груп співпадає з кількістю занять, які повинні провести викладачі.

Наявність потокових лекцій зменшує кількість занять, що проводять викладачі. Можна вважати, що у випадку потокової лекції декілька занять проводяться в одній аудиторії одним викладачем. Цей випадок є виключенням з правил складання розкладу занять, які говорять:

- на одній навчальній парів в одній аудиторії може бути призначене лише одне заняття;
- на одній навчальній парів один викладач може проводити лише одне заняття.

Розділення групи на підгрупи призводить до збільшення кількості занять, які проводять викладачі, та які потрібно розподілити по аудиторіях. При цьому кількість занять для студентів не змінюється. Таким чином, з одного заняття за навчальним планом виходить два заняття, які повинні проводити викладачі, та для яких потрібно призначати аудиторії для проведення. Розділення груп на підгрупи призводить не лише до збільшення кількості занять, що ставляться у розклад, воно призводить до ускладнення самого процесу складання розкладу занять, тому що потрібно паралельно поставити два заняття в одній групі в двох різних аудиторіях для двох викладачів.

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. В IDEF0 розглядаються логічні зв'язки між роботами, а не послідовність їх виконання в часі (WorkFlow). 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 [1 – 4].

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

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

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.

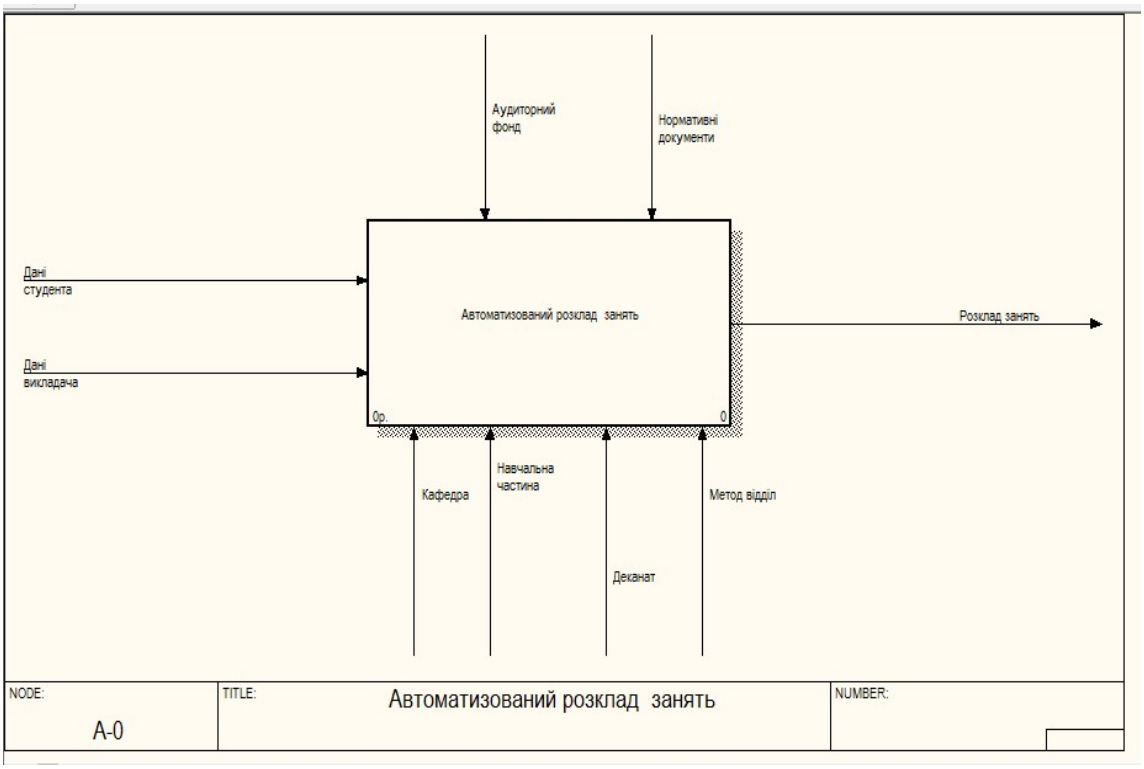

Рисунок 2.1 – Контекстна діаграма IDEF0

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».

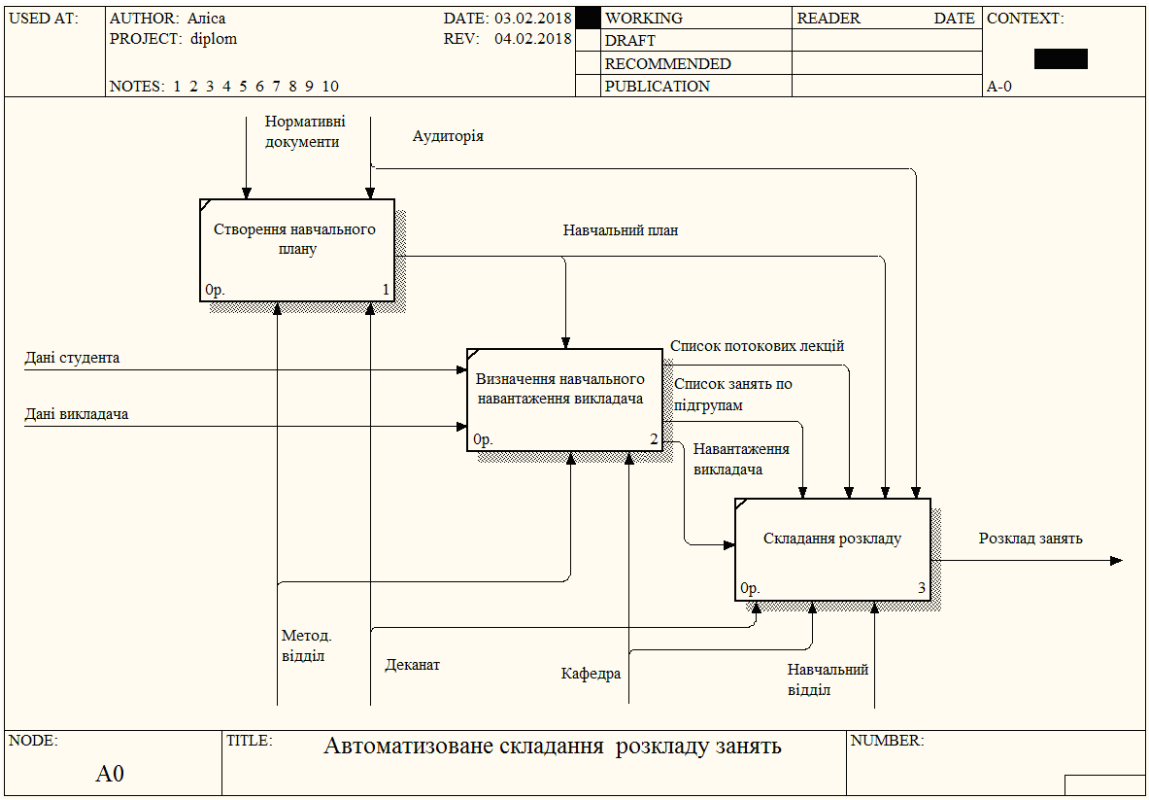

Рисунок 2.2 – Діаграма декомпозиції IDEF0

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

Наведена на рис. 2.2 діаграма декомпозиції IDEF0 враховує, що розрахунок навантаження викладачів на кафедрах змінює кількість занять викладачів за рахунок призначення потокових лекцій та лабораторних занять з розділенням на підгрупи.

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).

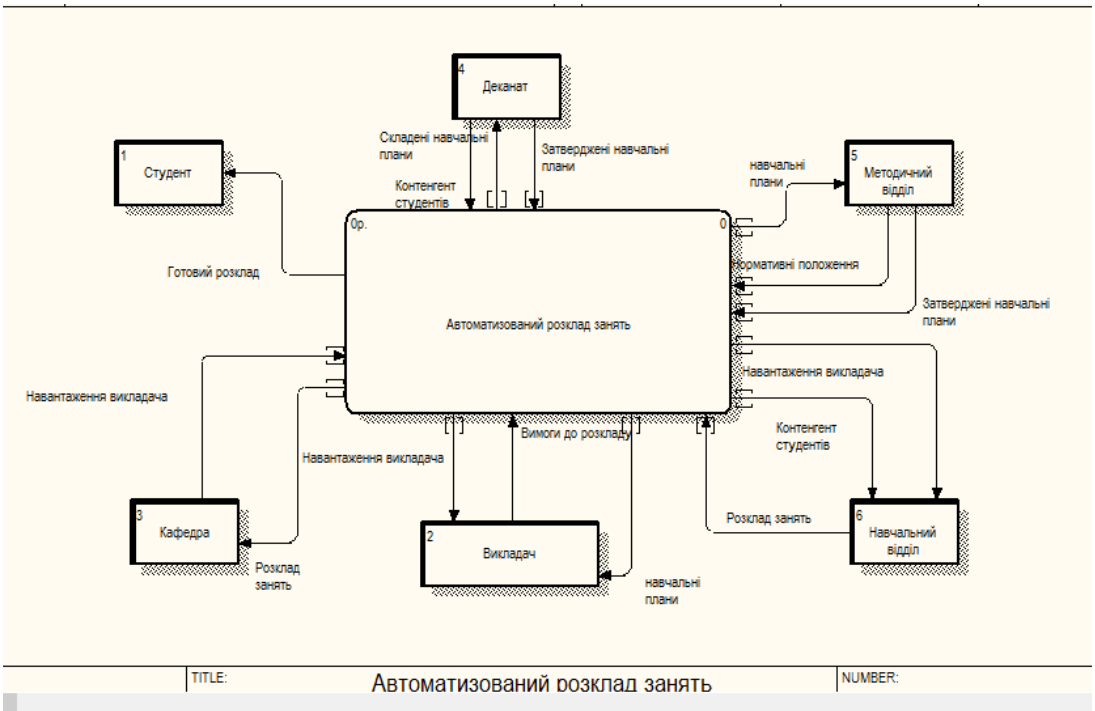

Рисунок 3.1 – Контекстна діаграма DFD автоматизованого розкладу занять

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 з урахуванням потокових лекцій та проведення занять у підгрупах.

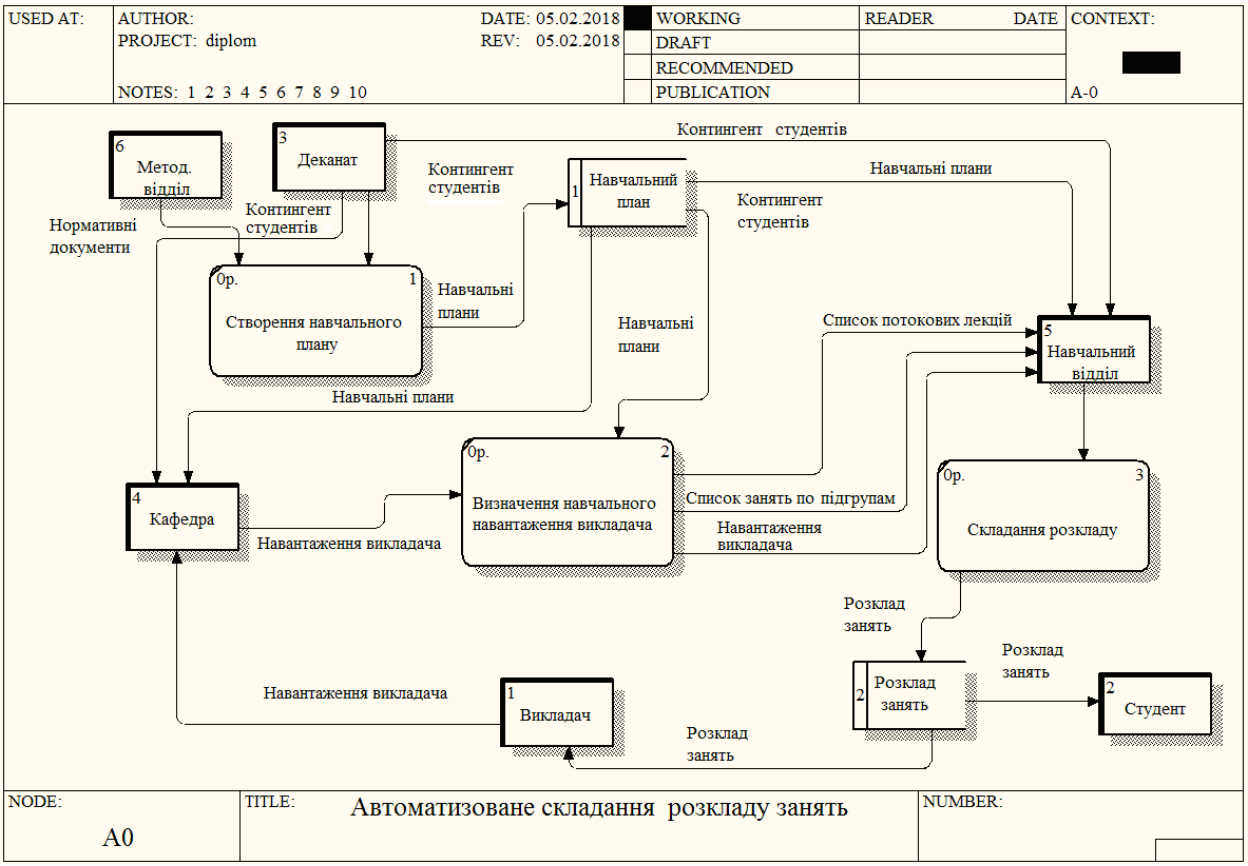

Рисунок 3.2 – Діаграма декомпозиції DFD автоматизованого розкладу занять

4 ФОРМУЛЮВАННЯ ЗАДАЧІ СКЛАДАННЯ РОЗКЛАДУ ЗАНЯТЬ В ТЕРМІНАХ АЛГЕБРИ ВІДНОШЕНЬ

4.1 Загальний випадок занять розміром однієї групи

У термінах алгебри відношень [5 – 7] добре формулюються процес складання розкладу занять.

Назвемо «заняттям» одну навчальну пару в одній академічній групі по одному предмету, що проводиться у одній аудиторії. Поки не будемо враховувати такі види 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 закладі вищої осиіти, відношення D містить список усіх предметів (навчальних дисциплін), що читаються в цьому закладі, 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 вишах студенти не вивчають усі можливі навчальні дисципліни, які викладаються у цьому ЗВО. Крім того не по всім навчальним дисциплінам передбачені всі види занять. Для деяких дисциплін передбачені лекції та семінари, для інших – лекції та лабораторні заняття. Можуть бути заплановані також заняття лише одного виду з деяких дисциплін.

Які навчальні дисципліни вивчають студенти, скільки годин відведено від вивчення кожної дисципліни, та які види занять передбачені для цієї дисципліни вказується у навчальних планах зі спеціальностей, за якими навчають у кожному ЗВО. Кожен факультет має набір навчальних планів, і кожна академічна група вчиться за деяким навчальним планом свого факультету.

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 (4.3):

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

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», тобто кожне заняття проводить один викладач, який може проводити багато занять.

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 прямого добутку над відношеннями S, A та P:

$$Z = S \times A \times P \qquad (4.4)$$

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:

$$Z = ((S \bowtie SA) \bowtie A) \times P \qquad (4.5)$$

Крім того, не всі заняття можуть проводитись на будь-якій навчальній парі тижня. Для академічних груп можуть призначатись дні для самостійної підготовки студентів, можуть призначатись дні тижня для отримання практичної або теоретичної підготовки за межами університету, тощо.

Деякі викладачі зайняті у деякі дні тижня, або на деяких навчальних парах виконанням інших робіт, не пов'язаних з проведенням занять. Крім того, у вишах можуть викладати фахівці, які працюють на постійній роботі у інших організаціях, а в ЗВО проводять лекції та практичні заняття з фахової підготовки студентів. Як правило, такі фахівці можуть викладати у закладі вищої оствіти лише у один конкретний день тижня, іноді вони можуть проводити заняття два дні на тиждень.

Крім того, програму автоматизованого складання розкладу занять можна використовувати для послідовного розрахунку розкладу у декілька кроків. На кожному кроці у розклад ставляться заняття для вибраної множини академічних груп ЗВО.

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:

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

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:

1) кожне заняття з відношення S повинне міститися у відношенні V і тільки один раз;
2) кожен викладач повинен мати рівно стільки годин занять в тиждень, скільки передбачено для нього за планом;
3) не може бути призначено більше за одне заняття для будь-якої групи на одну навчальну пару;
4) не може бути призначено більше за одне заняття для будь-якого викладача на одну навчальну пару;
5) 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.

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 потокових лекцій в задачі складання розкладу занять

Заняття академічних груп визначаються навчальними планами, за якими навчається група. Як видно з формули (4.2) кожне заняття має посилання на групу, дисципліну, вид занять, та навчальний план. Але

посилання на навчальний план для екземпляру сутності Заняття має тільки той сенс, що навчальний план для академічної групи визначає кількість годин для кожного виду занять по кожній дисципліні. Таким чином, після виділення з усієї множини годин навчання по дисципліні деякої кількості кожного виду занять з даної дисципліни, посилання на навчальний план у екземпляра сутності Заняття не потрібне.

Для врахування потокових лекцій потрібно для екземпляру сутності Заняття ввести посилання на інший тип сутності – Потік. Якщо заняття є лекцією, яка читається для заданого академічного потоку, то заняття буде мати позначку цього потоку. Для реалізації цієї можливості слід ввести атрибут «ідентифікатор потоку» у тип сутності Заняття. Для цього атрибуту слід допустити наявність пустих (порожніх) значень, які позначаються у базі даних як NULL. Значеннями NULL будуть відмічені всі заняття, які проводяться лише для однієї групи, а не для потоку груп.

Введення нового атрибуту у тип сутності Заняття дозволяє змінити формулювання 4-го та 5-го правила отримання реального розкладу занять з відношення Z (розділ 4.1):

1) не може бути призначено більше за одне заняття для будь-якого викладача на одну навчальну пару, якщо тільки ці заняття не є лекціями одного академічного потоку з одної дисципліни;
2) не може бути призначено більше за одне заняття для будь-якої аудиторії на одну навчальну пару, якщо тільки ці заняття не є лекціями одного академічного потоку з одної дисципліни.

Для потокових лекцій змінюється поняття допустимої навчальної пари для проведення заняття. У цьому випадку визначення допустимої пару можна сформулювати наступним чином: допустимою для проведення заняття є навчальна пара, яка вільна одночасно для викладача, для всіх груп, що належать до заданого академічного потоку, та хоча б для однієї допустимої для заняття аудиторії.

Таким чином у формулі 4.6 замість відношення PG повинне стояти відношення PF, яке визначає допустимі навчальні пари для академічного потоку.

Множина допустимих навчальних пар для потоку академічних груп буде перетинанням множин допустимих навчальних пар всіх груп, які входять у цей академічний потік.

4.3 Урахування занять з розділом груп на підгрупи в задачі складання розкладу занять

При обєднанні груп у потоки для читання лекцій у база даних не змінюється кількість занять, тобто кортежів у відношенні S. Відмінність від звичайних занять проявляється тільки при постановці потокової лекції в розклад: на відміну від звичайних занять у відношенні STT (Study in TimeTable) кілька кортежів можуть мати однакове поєднання номера навчальної пара з ідентифікатором аудиторії. Таким чином, потокова лекція розглядається як сукупність занять, які проводяться одночасно в одній аудиторії одним викладачем.

Але у випадку розділення групи на підгрупи з одного заняття за навчальним планом групи отримуємо два заняття викладачів, та два заняття для проведення у якій-небудь аудиторії. Тобто з одного екземпляру типу сутності Заняття_за_планом потрібно отримати два екземпляру типу сутності Заняття.

Такі перетворення можна виконати, якщо у тип сутності Заняття додати атрибут sg (subgroup – підгрупа). Цей атрибут може мати цілі значення, наприклад, 1 для підгрупи «а», 2 для підгрупи «б», 3 для всієї групи. Можна також використовувати логічний тип даних з можливістю значення NULL, яке буде відповідати випадку всієї групи.

Таким чином, при розділенні групи на підгрупи для проведення лабораторного заняття у базі даних замість одного кортежу відношення S з атрибутом sg = NULL, потрібно записати два кортежі, в яких атрибут sg буде мати значення 1 та 2 (для цілочисельного типу даних) або true (1) та false (0) – для булевого типу.

5 ОБЛІК ПОТОКІВ ТА ПІДГРУП У АЛГОРИТМІ СКЛАДАННЯ РОЗКЛАДУ ЗАНЯТЬ

5.1 Урахування потокових занять при складанні розкладу вручну

Наявність потокових лекцій ускладнює процес складання розкладу занять. Для постановки заняття у розклад потрібно вибирати навчальну пару, на якій вільні всі групи даного академічного потоку. При складанні розкладу занять вручну диспетчер зазвичай використовує аркуш великого розміру, на якому відображає розклад занять всіх груп, що можуть складати потік. Зазвичай потоки складаються з груп одного курсу одного факультету, і на одному аркуші паперу креслять таблицю для розкладу занять одного курсу одного факультету. У цьому випадку диспетчер може легко вибрати навчальну пару для проведення потокової лекції – вільний рядок таблиці розкладу для всіх груп потоку.

Іноді зустрічаються академічні потоки, що містять групи різних факультетів. Тоді диспетчеру потрібно покласти поряд декілька аркушів розкладу занять різних факультетів, щоб вибрати допустиму навчальну пару для постановки заняття у розклад.

Якщо диспетчер спочатку поставить у розклад семінари або лабораторні заняття, для постановки потокових лекцій може не знайтись місця – групи одного потоку можуть бути вільними на різних навчальних парах. Тому при складанні розкладу занять вручну зазвичай у розклад спочатку ставлять потокові лекції, особливо це відноситься до випадків потоків великого розміру та потоків з груп декількох факультетів.

Відступом від вказано правила складання розкладу занять є випадок наявності викладчів з дуже щільним розкладом занять, наприклад, викладачів з одним днем тижня, допустимим для проведення занять. У цьому випадку у розклад спочатку ставляться заняття вказаних викладачів, потім ставляться потокові лекції, а далі – всі інші заняття.

В алгоритмі автоматизованого складання розкладу занять з обліком потокових лекцій варто враховувати алгоритм складання розкладу занять вручну, принаймні потрібно намагатись ставити потокові лекції раніше постановки у розклад занять окремих груп для, щоб не зменшувати свободу розташування потокових лекцій у двомірному просторі (навчальна пара, аудиторія).

5.2 Урахування занять з поділом групи на підгрупи при складанні розкладу вручну

При складанні розкладу вручну зазвичай диспетчер намагається поставити на одну пару заняття обох підгруп. Якщо лабораторні заняття з дисципліни для заданої групи проводять два викладачі, та для них передбачено декілька допустимих аудиторій, диспетчер намагається на одну навчальну пару поставити лабораторні заняття з однієї дисципліни для обох підгруп. Цей алгоритм не завжди спрацьовує, оскільки для проведення всіх лабораторних занять з дисципліни може бути призначений лише один викладач, або лабораторні заняття з якоїсь дисципліни можуть проводитись лише в одній спеціалізованій лабораторії.

Якщо проведення лабораторної роботи з заданої дисципліни в обох підгрупах одночасно неможлива, диспетчер намагається знайти «парну» дисципліну для заданої дисципліни. У цьому випадку вибирається дві навчальні пари для постановки у розклад лабораторних занять для двох дисциплін: на одну пару ставляться лабораторна робота з першої дисципліни в підгрупі «а» та лабораторна робота з другої дисципліни у підгрупі «б»; на іншій парі – навпаки. Такий алгоритм можливий, якщо обидва викладачі, що ведуть лабораторні заняття з двох дисциплін, мають спільну множину допустимих навчальних пар для постановки занять у розклад, та обидві лабораторії для проведення занять з двох дисциплін також мають спільну множину вільних навчальних пар.

Якщо обидва описаних алгоритми постановки лабораторних занять з поділом групи на підгрупи неможливі, у розклад ставиться заняття однієї підгрупи. У цьому випадку диспетчер намагається для постановки заняття вибрати навчальну пару таким чином, щоб зменшити вірогідність виникнення «вікна» у розкладі другої підгрупи. Наприклад, четверту пару навчального дня тижня за наявності занять всієї групи на перших трьох парах. Або можна вибрати першу або четверту пару за наявності занять всієї групи на другій та третій парі.

У вільний від занять день не слід ставити заняття однієї підгрупи на третю пару, оскільки потрібно буде поставити заняття для усієї групи на третю пару та на четверту пару поставити заняття для всієї групи або іншої підгрупи. Перша пара у такому випадку буде вільна принаймні для другої підгрупи, що робить розклад занять не дуже комфортним для студентів.

У алгоритмі автоматизованого складання розкладу занять також слід передбачити можливість одночасної постановки занять обох підгруп академічної групи на одну навчальну пару. У разі неможливості або недоцільності такої постановки занять двох підгруп у розклад заняття однієї підгрупи слід ставити у розклад таким чином, щоб мінімізувати ймовірність появи «вікна» у розкладі другої підгрупи.

5.3 Визначення змінних коефіцієнтів сортування занять у списку

Як вказано у розділі 5.1, потокові лекції мають меншу свободу розташування у розкладі занять, тому бажано ставити їх у розклад до постановки занять окремих груп. Для відображення цієї ситуації у алгоритмі автоматизованого складання розкладу занять слід передбачити відповідний порядок сортування занять у списку для постановки їх у розклад. Потокова лекція має тим меншу свободу розташування, чим більша її розмірність, тобто кількість груп, що складають академічний потік. Тому до коефіцієнтів, за якими виконується сортування занять у списку, слід додати розмірність потоку Kg – кількість груп у потоці.

В першому розділі комплексної магістерської роботи визначені коефіцієнти, за якими слід упорядковувати заняття для постановки їх у розклад. Перші два коефіцієнти K1, K2 визначають свободу розташування кожного заняття у розкладі. Інші два коефіцієнти K3, K4 описують розклад занять викладача, що призначений для проведення заняття: K3 визначають свободу розташування у розкладі решти занять цього викладача (ще не поставлених у розклад), K4 – щільність решти занять у розкладі викладача.

При складанні розкладу занять вручну спочатку ставляться у розклад потокові лекції, якщо немає викладачів з дуже щільним розкладом занять. Тому при автоматизованому складанні розкладу занять можна прийняти наступний алгоритм:

– якщо існують викладачі з дуже щильним розкладом (K3<K3MAX), список занять упорядковується у наступному порядку:

1) за зростанням K3;
2) за спаданням K4;
3) за зростанням K1;
4) за зростанням K2;
5) за спаданням Kg;

– якщо немає викладачів з дуже щильним розкладом, список занять упорядковується у наступному порядку:

1) за зростанням K1;
2) за зростанням K2;
3) за спаданням Kg;
4) за зростанням K3;
5) за спаданням K4.

З визначенням коефіцієнту Kg для заняття підгрупи виникає деяка неоднозначність. З одного боку, занять можна поставити як на пару, що вільна для всієї підгрупи, так і на пару, що вільна тільки для цієї підгрупи, тобто заняття має більшу свободу розташування, ніж заняття цілої групи. З іншого боку, для зменшення ймовірності появи «вікна» у розкладі якоїсь підгрупи бажано ставити у розклад одночасно заняття обох підгруп, тобто такі заняття мають меншу свободу розташування, ніж заняття цілої групи. Тому для заняття підгрупи обираємо Kg = 1, як для заняття цілої групи.

5.3 Розрахунок допустимих пар для постановки у розклад потокової лекції

Потокову лекцію можна поставити у розклад, яка є одночасно вільною для всіх груп, які складают потік. Тобто множина вільних пар потоку є перетинанням множин вільних пар всіх груп потоку.

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). Зааняття ставиться на пари, на яких одночасно вільні група, викладач та акдиторія. А для випадку потокової лекції, для пари, яка вільна не для однієї групи, а для усій множини груп, що складають академічний потік.

Недоречно знаходити множини вільних пар кожного академічного потоку на кожному кроці розрахунку розкладу занять. Краще описати у базі даних новий асоціативний тип сутності FPF (Free Pare Flow – вільна пара потоку). Цей тип сутності буде мати лише два атрибути: ідентифікатор потоку та ідентифікатор навчальної пари.

Для розрахунку вільних пар академічного потоку на початку розрахунку розкладу занять можна спочатку занести в базу даних для кожного потоку кожну навчальну пару тижня. Потім для заданого потоку у циклі по групам даного потоку видалити з відношення FPF кортежі зі значенням ідентифікаторів навчальних пар, які не входять до множини вільних пар поточної групи з відношення FPG.

При постановці у розклад потокової лекції вибрана для неї навчальна пара усувается з переліку вільних пар цього потоку, всіх груп, що входять у цей потік, всіх інших потоків, до яких входять групи того потоку, заняття якого ставиться у розклад.

При постановці у розклад заняття однієї групи або підгрупи вибрана для заняття навчальна пара усувається зі списку вільних пар цієї групи, а також зі списків вільних пар всіх потоків, до яких входить дана група.

5.4 Розрахунок допустимих пар для постановки у розклад заняття для підгрупи

При постановці у розклад заняття підгрупи бажано знайти дві множини допустимих пар для заняття: множину допустимих пар для заняття, на яких вільна вся група, та множину допустимих пар для заняття, коли вільна тільки задана підгрупа.

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.

У випадку проведення занять без розділення груп на підгрупи асоціативний тип сутності FPG (Free Pare Group – вільні пари групи) повинен мати лише два атрибути: ідентифікатор групи та ідентифікатор навчальної пари, на якій вільна група. Але за наявності занять з поділом на групи можливі випадки, коли на якійсь парі заняття призначене лише одній підгрупі, а друга підгрупа вільна. Тобто, у відношенні FPG, як і у відношення Study, слід додати атрибут sg – позначку, що група вільна вся, або перша підгрупа, або друга підгрупа.

Таким чином, маємо набір відношень:

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

FPG (id_g, id_p, sg)

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_f – ідентифікатор потоку;
sg – позначка підгрупи;
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).

Тут SA (Study Auditory – аудиторії занять) – асоціативний тип сутності, що містить інформацію про те, які аудиторії є допустимими для проведення кожного заняття.

Нехай заняття, що потрібно поставити у розклад, має ідентифікатор id_s = I; ідентифікатор групи id_g = J; ідентифікатор викладача id_t = K; позначка підгрупи sg = SG (SG не дорівнює NULL).

Природне з'єднання відношень FPG і FPT містить навчальні пари, на яких кожен викладач може проводити заняття з кожною групою. Отже, щоб отримати навчальні пари, що вільні одночасно для всієї заданої групи та заданого викладача, потрібно зробити вибірки з відношень FPG і FPT, а потім виконати операцію природного з'єднання (5.1):

$$\text{Temp1}(\text{id\_t, id\_g, id\_p}) \leftarrow \sigma_{\text{id\_g} = j \text{ ^ sg is NULL}}(\text{FPG}) \bowtie \sigma_{\text{id\_t} = K}(\text{FPT}), \qquad (5.1)$$

Для отримання вільних навчальних пар для всіх аудиторій, допустимих для заданого заняття, потрібно виконати операцію природного з'єднання відношення SA з відношенням FPA – списком вільних пар аудиторій (5.2):

$$\text{Temp2}(\text{id\_st, id\_a, id\_p}) \leftarrow \sigma_{\text{id\_st} = I}(\text{SA}) \bowtie \text{FPA} \qquad (5.2)$$

Множина допустимих навчальних пар для заняття, коли вільна вся група, знаходиться за формулою (5.3):

$$Temp3\ (id\_p) \leftarrow \pi_{id\_p}(Temp1 \rhd\lhd\ Temp2) \qquad (5.4)$$

Щоб отримати допустимі для заняття з ідентифікатором id_s = I навчальні пари, на яких вільна тільки підгрупи, заняття якої потрібно поставити у розклад, потрібно вибрати навчальні пару, на яких вільна тільки ця підгрупа (5.5) – (5.6):

$$Temp4(id\_t, id\_g, id\_p) \leftarrow \sigma_{id\_g = j\hat{}\ sg = SG}(FPG) \rhd\lhd\ \ \sigma_{id\_t = K}(FPT), \qquad (5.5)$$

$$Temp5\ (id\_p) \leftarrow \pi_{id\_p}(Temp4 \rhd\lhd\ Temp2) \qquad (5.6)$$

Множини вільних для заняття пар Temp3 та Temp5 не мають спільних елементів, тому що у відношенні FPG для однієї групи не може бути більше одного кортежу для будь-якого ідентифікатора навчальної пари id_p. На вільній парі група може бути вільна або уся, або якась одна з підгруп.

Сумарна кількість елементів у множинах Temp3 та Temp5 відповідає коефіцієнту K1 для заняття підгрупи, який використовується для упорядкування занять в списку для постановки у розклад.

5.5 Загальний алгоритм складання розкладу занять

На начальному етапі складання розкладу занять коефіцієнти K1 та K2 великі для усіх занять. Тому сортування занять у списку для розкладу буде вибиратись в першу чергу за іншими критеріями. Якщо ніхто з викладачів, які ведуть заняття, немає дуже щільного розкладу занять (з малою різницею між кількістю допустимих для занять пар викладача та кількостю самих занять), то заняття у першу чергу будуть упорядковуватися за спаданням розмірності потоку, тобто в першу чергу у розклад будуть ставитись потокові лекції для великої кількості груп.

У розклад ставиться заняття, яке опинилось першим у списку. Для нього спочатку для усіх можливих для постановки заняття у розклад навчальних пар розраховуються пріоритетом пріоритету, які визначають, наскільки постановка заняття на дану навчальну пару буде відповідати оптимальному розкладу занять групи. Якщо заняття є потоковою лекцією, то для кожної групи потоку окремо розраховуються пріоритети всіх навчальних пар, допустимих для постановки у розклад лекції.

З допустимих навчальних пар викреслюються навчальні пари, котрі для якої-небудь групи отримали мінімальний пріоритет, наприклад, створюють для цієї групи подвійне «вікно» у розкладі занять. Для інших допустимих пар знаходиться сумарний пріоритет по всіх групах потоку.

Знаходяться пріоритети допустимих пар для викладача, що веде заняття.

Допустимі пари упорядковуються за сумарним пріоритетом групи (потоку) та викладача, при цьому приорітет викладача входить у суму з меншим ваговим коефіцієнтом.

Якщо на навчальну пару ставляться у розклад заняття двох підгруп одночасно, то підсумовуються пріоритети двох викладачів.

Для постановки заняття у розклад вибирається пара з максимальним сумарним пріоритетом.

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 (Study in TimeTable – заняття у розкладі) 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. Також кортежі з відповідним значенням навчальної пари видаляються з відношення FPF для всіх потоків, до яких входить група, заняття якої поставлене у розклад.

Якщо у розклад поставлена потокова лекція, то кортежі з відповідним значенням навчальної пари видаляються з відношення FPG – для всіх груп, що входять у потік, з відношення FPF – для даного потоку та для всіх потоків, до яких входить групи, потокова лекція для яких поставлена у розклад.

Якщо у розклад поставлене заняття підгрупи, то кортеж з відношення FPG для цієї групи видаляється, якщо навчальна пара вже була вільна лише для цієї підгрупи. Якщо пара була вільна для всієї підгрупи, то кортеж з заданим значенням навчальної пари для згаданої групи з відношення FPG не видаляється, а для нього ставиться відмітка, що навчальна пара вільна для

протилежної групи. Але з відношення FPF для всіх потоків, до яких входить група, видаляються кортежі з відповідним значенням навчальної пари.

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.

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».

Відкриті публікації останніх років не містять моделі та алгоритму врахування потокових лекцій та паралельних занять з поділом груп на підгрупи [8 – 14]. Платні програмні продукти пропонуються з можливістю урахування цих факторів при складанні розкладу занять, але вони не мають відкритого алгоритму та коду. Отже, порівняння запропонованого алгоритму з наявними неможливе.

6 АЛГОРИТМ ВИЗНАЧЕННЯ ОПТИМАЛЬНОЇ НАВЧАЛЬНОЇ ПАРИ ДЛЯ ПРОВЕДЕННЯ ЗАНЯТТЯ

6.1 Розрахунок пріоритетів вільних навчальних пар для постановки у розклад потокових лекцій

Алгоритм додавання заняття до розкладу повинен передбачати виконання декількох умов, які забезпечують комфортний розклад занять студентів та викладачів. У розкладі занять академічних груп неприпустима наявність «вікон», або однієї пари протягом навчального дня. Для розкладів занять викладачів допускається наявність «вікон» або одиночних пар, але потрібно мінімізувати їх кількість.

Розрахунок «вікон» у розкладі занять групи або викладача виконується дуже просто при використанні мови запитів SQL – мови реляційних баз даних [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 групи. «Вікно» у розкладі занять є в той день, коли:

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

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.

Для потокової лекції не потрібно розробляти окремий алгоритм призначення пріоритетів навчальних пар. Як сказано у розділі 5.5, пріоритети допустимих для заняття павчальних пар розраховуються окремо для кожної групи, що входять у академічний потік.

Серед допустимих навчальних пар для проведення лекції знаходяться ті, що мають занизький пріоритет для якої-небудь групи потоку, наприклад, у випадку, коли постановка заняття на вибрану пару призведе до появи «вікна» у розкладі занять групи. Якщо навчальних пар з незадовільним пріоритетом окремих груп потоку менше половини всіх наявних допустимих пар для постановки у розклад потокової лекції, то ці пари видаляються з множини допустимих навчальних пар для лекції. Решта допустимих навчальних пар упорядковується за сумарним пріоритетом всіх груп потоку та викладача, та вибирається пара з максимальним сумарним пріоритетом.

Пошук навчальних пар для оптимального розташування у розкладі занять окремих груп розглядається у першій частині комплексної магістерської роботи.

6.2 Розрахунок пріоритетів вільних навчальних пар для постановки у розклад занять з розділенням на підгрупи

Для заняття підгрупи потрібно знайти дві множини допустимих пар для постановки у розклад: множину пар Temp3, коли вільна вся група (формула 5.4), та множину пар Temp5, коли вільна тільки задана група (формула 5.6).

Для навчальних пар з множини Temp5 розраховуються пріоритети навчальних пар за наступним алгоритмом:

1) максимальний пріоритет (1) призначається парі, яка є «вікном» у розкладі занять підгрупи, також максимальний пріоритет призначається другій парі для дня, коли у заданої підгрупи є тільки перша пара, а у другої підгрупи – дві пари, або третій парі для дня, коли у заданої підгрупи є тільки четверта пара, а у другої підгрупи – третя та четверта пари; також максимальний пріоритет призначається для 2-ї пари, якщо у заданої пари немає занять у цей день, в у іншої підгрупи є тільки друга пара;
2) наступний пріоритет (2) призначається для пари з номером Pmin – 1, якщо в цей день у заданої підгрупи є тільки друга та третя пари;
3) наступний пріоритет (3) призначається для третьої пари, якщо в цей день у розкладі підгрупи є тільки одна друга пара; також пріоритет 2 призначається для 2-ї пари, якщо у заданої підгрупи немає занять у цей день, в у іншої підгрупи є тільки друга пара;
4) 4-й пріоритет призначається для четвертої пари, якщо в цей день у розкладі підгрупи є тільки одна третя пара;

5) 5-й пріоритет призначається для пари з номером Pmin – 1, якщо в цей день у підгрупи 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 у підгрупи є дві пари поспіль – 1 та 2 або 2 та 3;
7) 8-й пріоритет призначається першій парі у день, коли у підгрупи є друга та четверта пари;
8) 9-й та 10-й пріоритети призначаються для 1-ї та 3-ї пари 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.

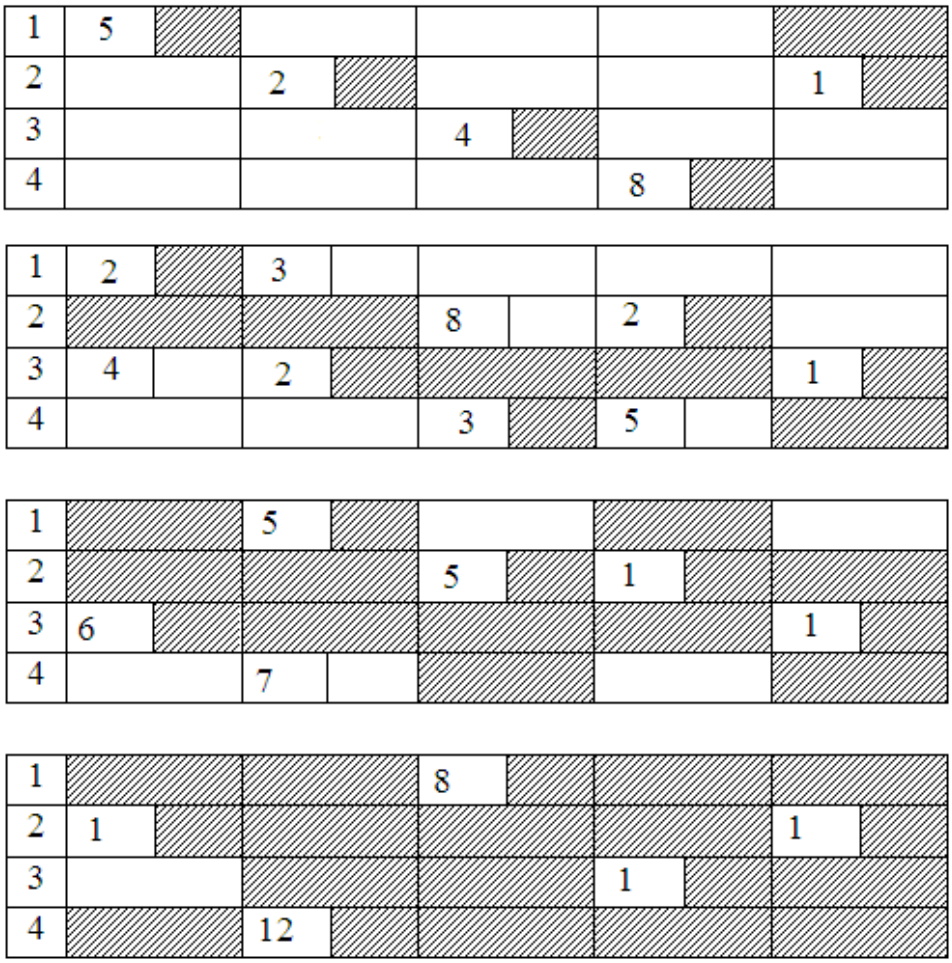

Рисунок 6.1 – Пріоритети для вибору вільної пари для заняття підгрупи

Для допустимих пар з множини Temp3 потрібно визначити, чиможна поставити на вільну пару одночасно заняття для обох груп. Для цього треба

вибрати заняття другої підгрупи, які можуть проводитись на множині пар Temp3, та мають іншого викладача і можуть проводитись у іншій аудиторії. Ця вибірка виконується за формулами (6.2) – (6.4).

Формула (6.2) розраховує спільний список вільних пар групи та інших викладачів:

$$Temp6(id\_t, id\_p) \leftarrow \sigma_{id\_g = j\text{^}\ sg\ != SG}(FPG) \rhd\lhd \sigma_{id\_t\ != K}(FPT), \qquad (6.2)$$

Формула (6.3) розраховує занять іншої підгрупи з допустимими для них аудиторіями та вільними парами для цих аудиторій:

$$Temp7\ (id\_st,id\_t,id\_a,id\_p) \leftarrow (\sigma_{id\_g = j\text{^}\ sg\ != SG}(Study) \rhd\lhd SA) \rhd\lhd FPA \qquad (6.3)$$

Список занять іншої підгрупи, що можуть проводитись на множині пар Temp3 с закріпленими за ними викладачами та з допустимими для проведення аудиторіями знаходиться за формулою (6.4):

$$Temp8\ (id\_st, id\_t, id\_a, id\_p) \leftarrow Temp7 \rhd\lhd Temp6 \rhd\lhd Temp3 \qquad (6.4)$$

Якщо відношення Temp8 не порожнє, то можна поставити у розклад обидві підгрупи одночасно. Пріоритети вільних пар для постановки занять обох підгруп призначаються як для постановки у розклад заняття цілої групи (рис. 6.2). Але для розрахунку загального пріоритету вільних пар для постановки у розклад занять обох підгруп потрібно враховувати сумарний пріоритет групи та двох викладачів.

У множину Temp3 може входити досить велика кількість навчальних пар, особливо на початку розрахунку розкладу занять. Також для кожної з цих пар можуть знайтись заняття декількох викладачів. Для зменшення кількості розрахунків пріоритетів для великої кількості пар у декількох викладачів бажано спочатку вибрати кілька допустимих пар з максимальним пріоритетом для групи, і вже для цієї підмножини допустимих пар розраховувати пріоритети навчальних пар для викладачів.

Якщо відношення Temp8 порожнє, але множина навчальних пар Temp3 не порожня, то існує можливість поставити на вільну пару групи заняття тільки однієї підгрупи. У цьому випадку пріоритету вільних пар розраховуються за умови зменшення ймовірності отримання «вікна» у розкладі занять іншої підгрупи та всієї групи (рис.6.3).

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 6.2 – Пріоритети для вибору вільної пари для заняття групи

1	9			5		7		14	
2	13		10			10		8	
3	10		9		4			10	
4	15		17		11		3		

1		3		10			10	
2				8		20		
3	6						20	
4	16		6			12		

1				12				
2	20						20	
3	20					20		
4		14						

Рисунок 6.3 – Пріоритети для вибору вільної пари для заняття групи на вільній парі всієї групи

6.3 Урахування занять через тиждень у розрахунку пріоритетів вільних навчальних пар для підгрупи

Наявність у розкладі занять через тиждень значно ускладнює алгоритм розрахунку оптимальної навчальної пари для постановки заняття у розклад, особливи для випадку занять з розділенням груп на підгрупи. Для потокових лекцій не потрібно розробляти окремий алгоритм урахування занять через тиждень, тому що для окремих груп алгоритм урахування занять через тиждень розроблений у першій частині комплексної магістерської роботи.

Загальний випадок постановки у розклад заняття підгрупи через тиждень має дуже багато варіантів. Тому можна в програмі автоматизованого складання розкладу занять ставити занять через тиждень тільки для обох підгруп разом за алгоритмом постановки через тиждень заняття цілої групи.

7 СИСТЕМНИЙ АНАЛІЗ І КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ БАЗИ ДАНИХ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Для використання програми автоматизованого складання розкладу занять потрібно розробити базу даних інформаційної системи «Навчальний 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.

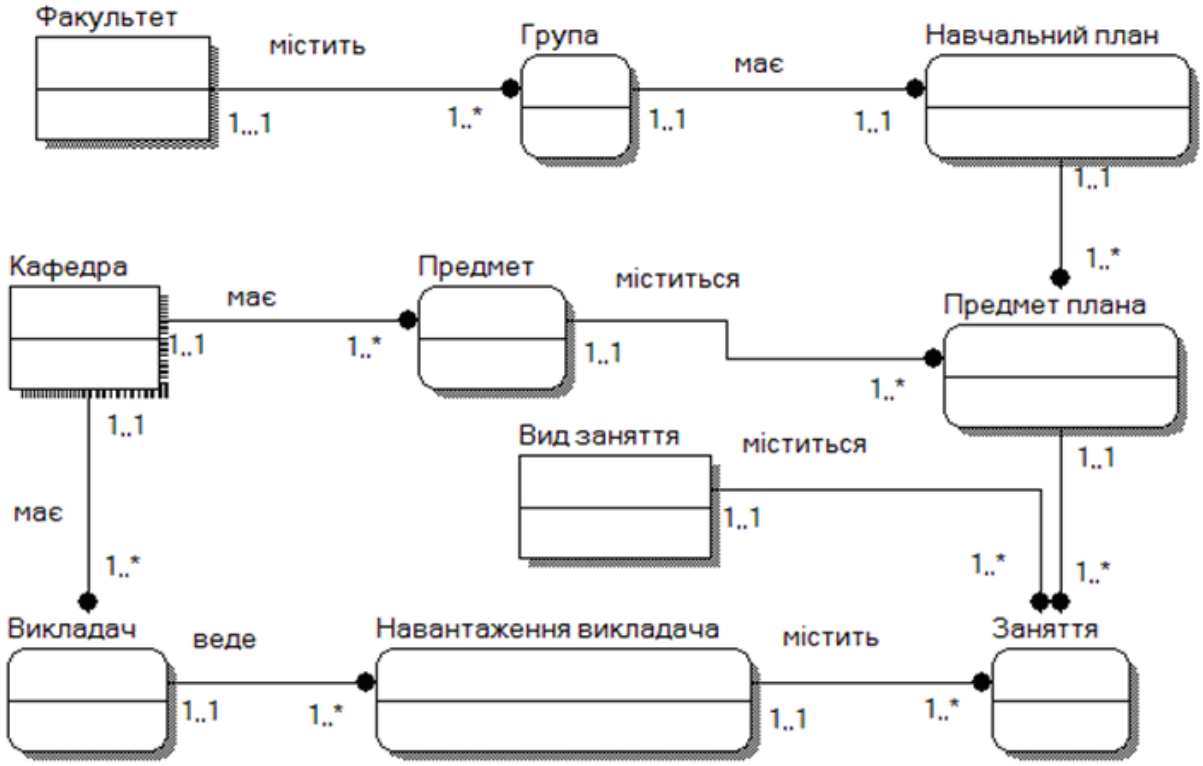

Рисунок 7.1 – Перша ER-діаграма бази даних «Розклад занять»

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.

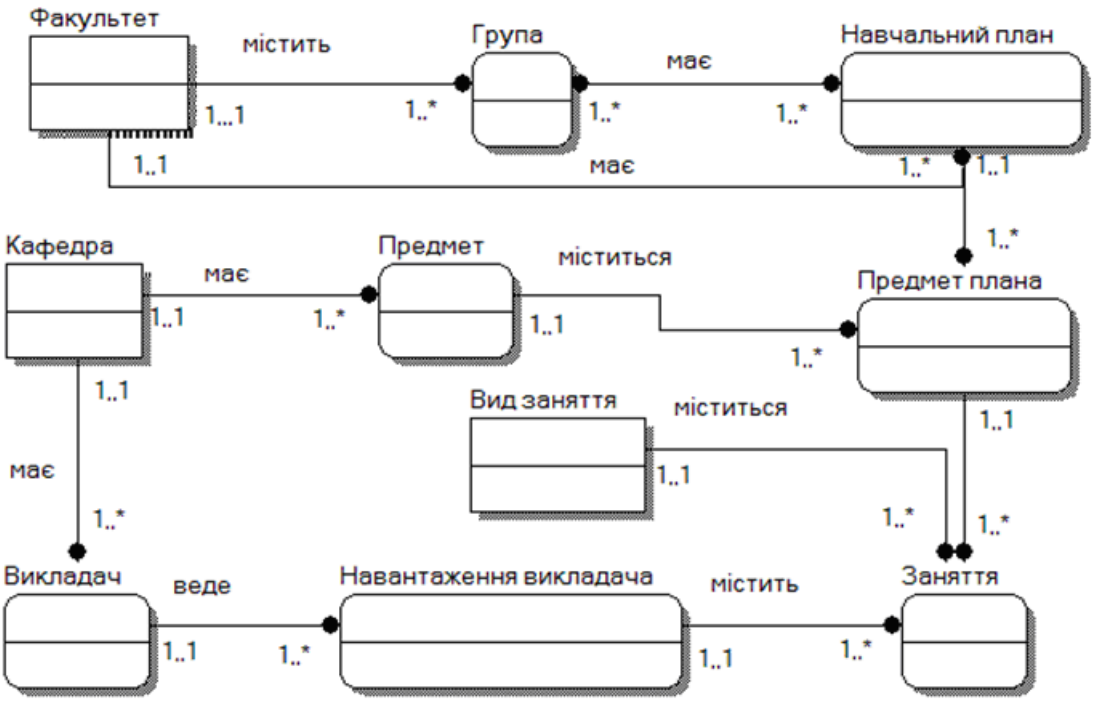

Рисунок 7.2 – Друга ER-діаграма бази даних «Розклад занять»

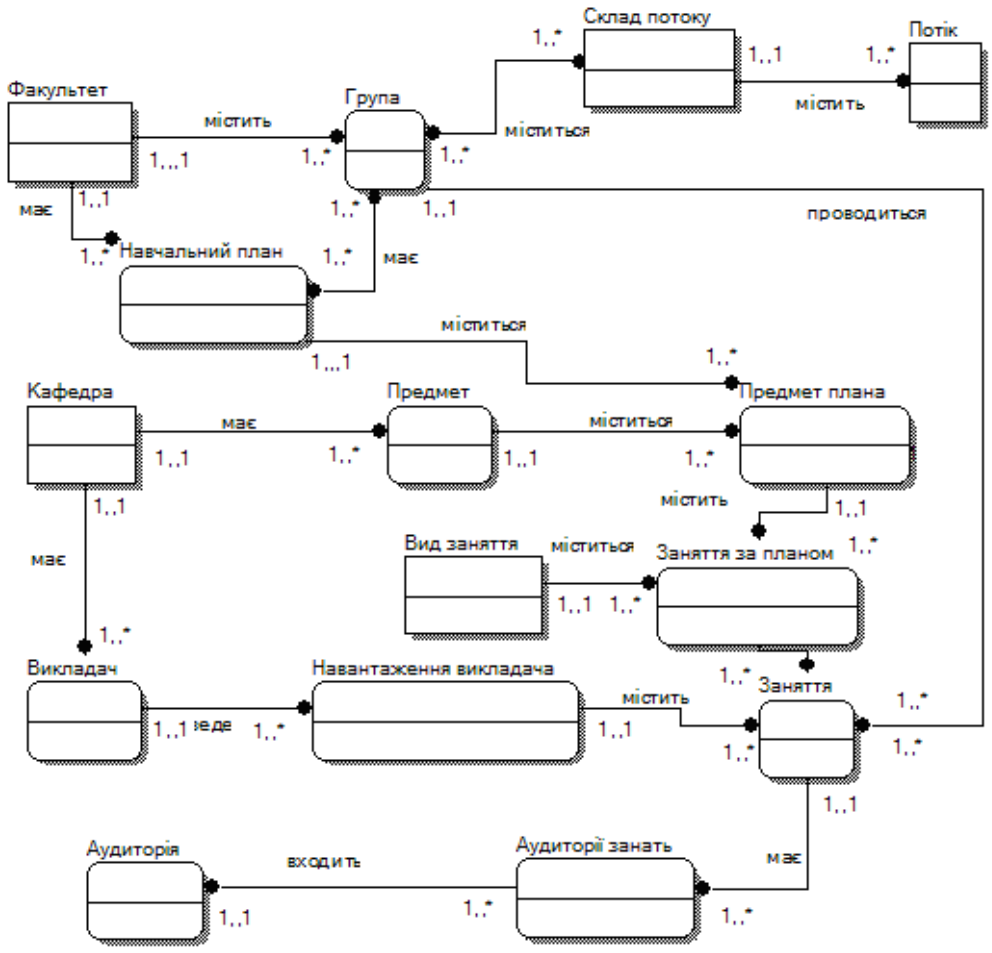

Рисунок 7.3 – Доповнена ER-діаграма БД «Розклад занять»

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).

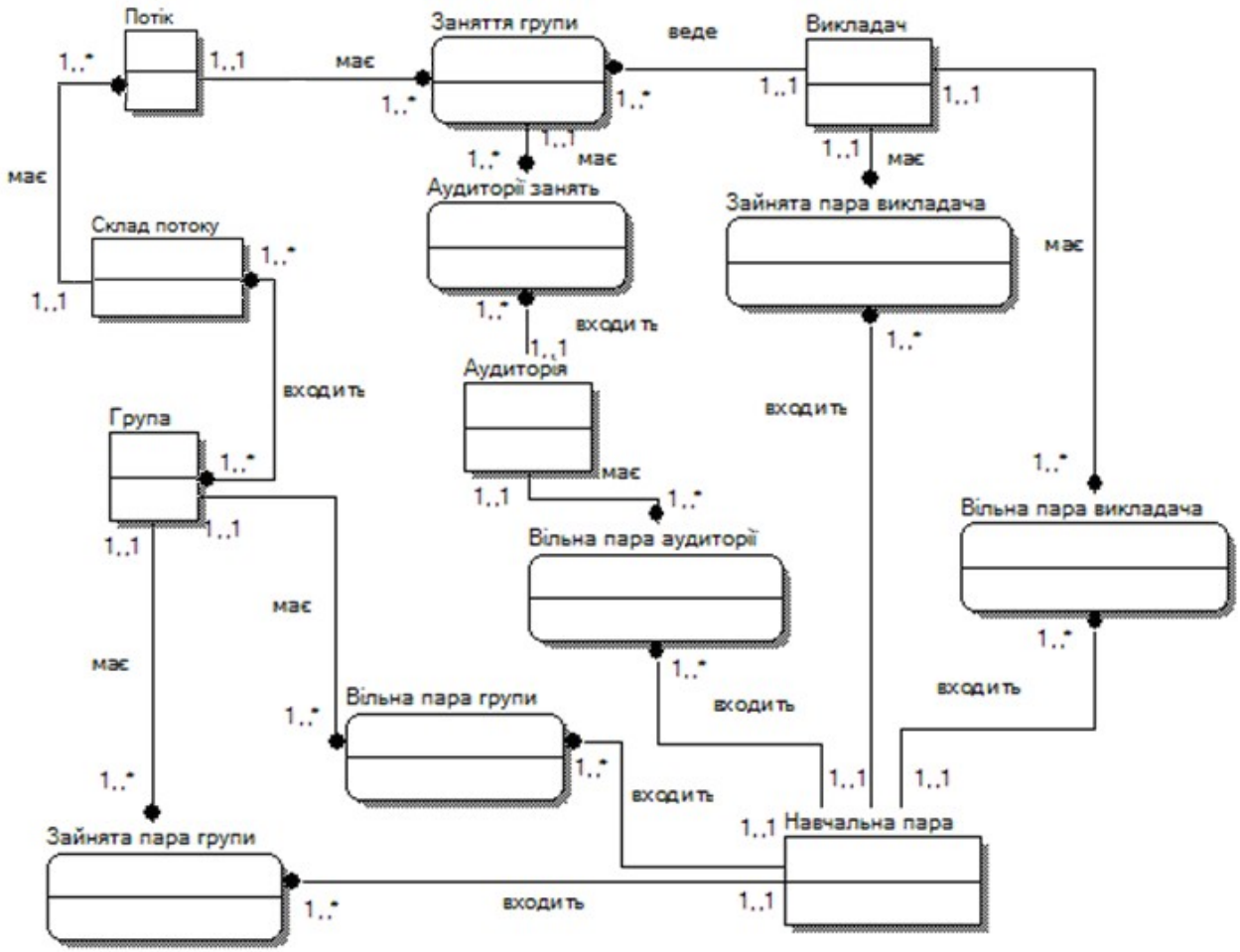

Рисунок 7.4 – ER-діаграма БД «Розклад занять»:
підсхема «Розрахунок розкладу»

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.

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.

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).

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 .

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

З порівняння різних редакцій MS SQL 2005 видно, що кожна з них має свої переваги. Версія Express має менш за всіх набір функцій, але вона безкоштовна, проста в експлуатації і призначена для створення простих додатків управління даними, на відміну від версії Enterprise, яка являє собою комплексну платформу для даних і аналізу, орієнтована на критично важливі бізнес-додатки. Вона підтримує більший спектр функцій аніж інші редакції MS SQL 2005.

Також, якщо порівнювати MS SQL 2005 та MS SQL 2008, то могна було б використовувати версію 2008, вона включає корисні удосконалення: нові типи даних, підтримку механізму налагодження кодів T-SQL і автоматичного заповнення IntelliSense, а також поліпшення в механізмах бізнес-аналітики (BI). Але враховуючи, що вже багато років у університеті при розробці системи автоматичного складання розкладу використовується MS SQL 2005, перехід на MS SQL 2008 може повести за собою ряд наслідків. Наприклад, обробник запитів може вести себе некоректно і видавати помилки, тому всі запити, які проходять у версії MS SQL 2005 необхідно буде переписувати під версію MS SQL 2008, що є небажанним.

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 реляційній алгебрі – 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].

Розрахунок розкладу занять виконується за допомогою збереженої процедури MakeTimeTable, розробленої у першій частині для постановки у розклад занять однієї групи – щотижневих та через тиждень. Ця процедура має декілька параметрів, що визначають її роботу.

Диспетчер навчального відділу повинен мати клієнтський додаток для роботи з програмою автоматизованого складання розкладу занять. За допомогою цього застосування користувач 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».

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.

В даній частині комплексної роботи були внесені зміни у код процедури MakeTimeTable.

Для розрахунку пріоритетов вільних пар академічного потоку викликається збережена процедура FillFlowParePriorWeek, яка знаходить пріоритети вільних пар для постановки у розклад як щотижневої лекції, так і лекції через тиждень.

Додано програмний код для вибору у двомірному просторі (навчальна пара, аудиторія) оптимального місця для постановки у розклад щотижневої потокової лекції, а також додано програмний код для вибору у тримірному просторі (навчальна пара, аудиторія. тиждень) оптимального місця для постановки у розклад потокової лекції через тиждень.

Цей код наведено на рис. 9.1.

Для розрахунку пріоритетів вільних пар окремої підгрупи у процедурі MakeTimeTable викликається збережена процедура FillSubGrParePrior. Вона розраховує пріоритети для постановки у розклад щотижневого заняття однієї підгрупи академічної групи.

Для постановки у розклад щотижневої потокової лекції у процедурі MakeTimeTable викликається збережена процедура Study2TTEveryWeekFlow, для постановки у розклад лекції через тиждень – Study2TTWeekFlow.

```
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
```

Рисунок 9.1 – Програмний код вибору навчальної пари та аудиторії для постановки у розклад потокової лекції

Для постановки у розклад щотижневого заняття однієї підгрупи викликається збережена процедура Study2TTEveryWeekSubGr.

Процедура FillFlowParePriorWeek розраховує пріоритети вільних пар потоку для постановки заняття у його розклад:

CREATE proc FillFlowParePriorWeek

@id_study int,

@id_fl smallint,

@week bit

Параметри процедури:

@id_study – ідентифікатор заняття;

@id_fl – ідентифікатор академічного потоку;

@week – позначка, чи проводиться заняття кожного тижня.

Процедура FillFlowParePriorWeek вибіркою з відношень Study та Flow-Group розраховує ідентифікатори груп, що складають потік з ідентифікатором id_fl= @id_fl для заняття з ідентифікатором id_study= @id_study. Також розраховуються ідентифікатори занять, під якими для кожної групи записана ця потокова лекція. Для кожної групи викликається процедура FillGrParePriorWeek, яка розраховує пріоритети вільних пар групи, допустимих для постановки лекції у розклад даної лекції.

Знаходяться вільні пари, на яких для яких-небудь груп призначається замалий пріоритет, і ці пари виключаються з підрахунку сумарного пріоритету навчальних пар.

Процедура FillSubGrParePrior розраховує пріоритети вільних пар для випадку постановки у розклад щотижневого заняття однієї підгрупи.

CREATE proc FillSubGrParePrior
@id_study int,
@id_gr smallint,
@subgr bit

Параметри процедури:

@id_study – ідентифікатор заняття, що ставиться у розклад;

@id_gr – ідентифікатор групи;

@subgr – позначка пфдгрупи.

Процедура FillSubGrParePrior вибіркою з відношень Study, STT заповнює тимчасові таблиці GrParePrior та GrSubParePrior. Таблиця GrParePrior для кожної навчальної пари тижня містить відмітку, є на цій парі заняття у групи, чи немає. Таблиця GrSubParePrior містить аналогічні дані, але окремо для кожної підгрупи.

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

У циклі по днях навчального тижня для кожного дня тижня викликаються процедури розрахунку пріоритетів для постановки заняття підгрупи на вільних у цей день навчальних парах групи DoPriorGrSub та вільних парах у підгрупі DoPriorSubGr.

Процедури DoPriorGrSub та DoPriorSubGr розраховують пріоритети для різних множин навчальних пар. Після виклаку цих процедур пріоритети об’єднаної множини вільних пар записуються у таблицю GrParePrior.

Процедура Study2TTEveryWeekFlow ставить у розклад щотижневу потокову лекцію:

CREATE proc Study2TTEveryWeekFlow
@id_study int,
@id_gr smallint,
@id_teach smallint,
@id_fl smallint,
@id_pare smallint,

@id_aud smallint

Параметри процедури:

@id_study – ідентифікатор заняття;

@id_gr – ідентифікатор академічної групи;

@id_teach – ідентифікатор викладача;

@id_fl – ідентифікатор академічного потоку ;

@id_pare – ідентифікатор навчальної пари;

@id_aud – ідентифікатор аудиторії.

Процедура Study2TTEveryWeekFlow вибіркою з відношень Study та FlowGroup розраховує ідентифікатори груп @@id_gr, що складають потік з ідентифікатором id_fl= @id_fl для заняття з ідентифікатором id_study= @id_study. Також розраховуються ідентифікатори занять @@id_study, під якими для кожної групи записана ця потокова лекція.

Для кожного заняття (лекції) кожної групи процедура 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 Study2TTWeekFlow ставить у розклад потокову лекцію, що проводиться через тиждень:

CREATE proc StudyToTTWeek

@id_study int,

@id_gr smallint,

@id_teach smallint,

@id_fl 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 Study2TTEveryWeekFlow.

Процедура Study2TTEveryWeekFlow вибіркою з відношень Study та FlowGroup розраховує ідентифікатори груп @@id_gr, що складають потік з ідентифікатором id_fl= @id_fl для заняття з ідентифікатором id_study= @id_study. Також розраховуються ідентифікатори занять @@id_study, під якими для кожної групи записана ця потокова лекція.

Для кожного заняття (лекції) кожної групи процедура 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 кожної 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.

Study2TTEveryWeekSubGr

Процедура Study2TTEveryWeekSubGr ставить у розклад щотижневу заняття підгрупи:

```
CREATE  proc Study2TTEveryWeekSubGr
@id_study int,
@id_gr smallint,
@id_teach smallint,
@subgr bit,
@id_pare smallint,
@id_aud smallint
```

Параметри процедури:

@id_study – ідентифікатор заняття;

@id_gr – ідентифікатор академічної групи;

@id_teach – ідентифікатор викладача;

@subgr – позначка підгрупи;

@id_pare – ідентифікатор навчальної пари;

@id_aud – ідентифікатор аудиторії.

Процедура Study2TTEveryWeekSubGr 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 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.

З таблиці вільних пар групи видаляються кортежі зі значенням атрибутів: id_gr = @id_gr, id_pare = @id_pare, subgr = @subgr. Якщо немає такого кортежу, то у кортежі з атрибутами id_gr = @id_gr, id_pare = @id_pare, subgr = NULL значення атрибуту subgr змінюється на ! @subgr.

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 групи з ідентифікатором id_g = J знаходиться вибіркою зі зв’язаних таблиць Study та STT:

$$TTg\ (id\_st, id\_sub, id\_kind, id\_t, id\_p, id\_a) \leftarrow (\sigma_{id\_g=J}(Study)) \bowtie STT \qquad (9.1)$$

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

1-й стовпчик – номер пари протягом тижня, наприклад, «Пн-1»;

2-й стовпчик – дані про заняття групи на цій парі у вигляді одного рядка тексту.

Вибірку занять розкладу слід доповнити порожніми парами, коли у групи 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. Якщо у групи є заняття з розділенням на підгрупи, то потрібно у окремі стовпчики виводити заняття кожної підгрупи.

Приклад розкладу занять заданої групи на одному тижні наведено на рис. 9.1.

Пара	нечетная неделя, подгр."а"	нечетная неделя, подгр."б"
Пн-1		
Пн-2		
Пн-3		
Пн-4		
Пн-5		
Вт-1	Фізика лаб.раб,512(2) Петров П.П.	
Вт-2	Фізика лекция,510(2) Петров П.П.	Фізика лекция,510(2) Петров П.П.
Вт-3	ООМ лекция,242(1) Чмырь И.А.	ООМ лекция,242(1) Чмырь И.А.
Вт-4	ВишМат_ММДО лекция,409(1) Глушков А.В.	ВишМат_ММДО лекция,409(1) Глушков А.В.
Вт-5		
Ср-1	ВишМат_ММДО лекция,409(1) Глушков А.В.	ВишМат_ММДО лекция,409(1) Глушков А.В.
Ср-2	Числ.мет лекция,241(1) Крыжановская Т.В.	Числ.мет лекция,241(1) Крыжановская Т.В.
Ср-3		ВишМат_ММДО лаб.раб,318(1) Буяджи В.И.
Ср-4		
Ср-5		
Чт-1	Електротехніка лекция,242(1) Горьев С.А.	Електротехніка лекция,242(1) Горьев С.А.
Чт-2	ин.яз	ин.яз
Чт-3	физкультура	физкультура
Чт-4		
Чт-5		
Пт-1	ВишМат_ММДО лаб.раб,318(1) Буяджи В.И.	Електротехніка лаб.раб,126(1) Горьев С.А.
Пт-2	Числ.мет лаб.раб,324(1) Крыжановская Т.В.	Числ.мет лаб.раб,327б(1) Шпинарева И.М.
Пт-3	Електротехніка лаб.раб,126(1) Горьев С.А.	Фізика лаб.раб,512(2) Петров П.П.
Пт-4		
Пт-5		
Сб-1		
Сб-2		
Сб-3		
Сб-4		
Сб-5		

Рисунок 9.1 – Розклад занять групи на парному тижні

Повний розклад занять зананої групи на обох тижнях наводиться на рис. 9.2.

Пара	нечетная неделя, подгр."а"	нечетная неделя, подгр."б"	четная неделя, подгр."а"	четная неделя, подгр."б"
Пн-1				
Пн-2				
Пн-3				
Пн-4				
Пн-5				
Вт-1	Фізика лаб.раб,512(2) Петров П.П.		ВишМат_ММДО семинар,310(1) Глушков А.В.	ВишМат_ММДО семинар,310(1) Глушков А.В.
Вт-2	Фізика лекция,510(2) Петров П.П.	Фізика лекция,510(2) Петров П.П.	Фізика лекция,510(2) Петров П.П.	Фізика лекция,510(2) Петров П.П.
Вт-3	ООМ лекция,242(1) Чмырь И.А.	ООМ лекция,242(1) Чмырь И.А.	ООМ лекция,242(1) Чмырь И.А.	ООМ лекция,242(1) Чмырь И.А.
Вт-4	ВишМат_ММДО лекция,409(1) Глушков А.В.	ВишМат_ММДО лекция,409(1) Глушков А.В.	ООМ семинар,321(1) Чмырь И.А.	ООМ семинар,321(1) Чмырь И.А.
Вт-5				
Ср-1	ВишМат_ММДО лекция,409(1) Глушков А.В.	ВишМат_ММДО лекция,409(1) Глушков А.В.	ВишМат_ММДО лекция,409(1) Глушков А.В.	ВишМат_ММДО лекция,409(1) Глушков А.В.
Ср-2	Числ.мет лекция,241(1) Крыжановская Т.В.	Числ.мет лекция,241(1) Крыжановская Т.В.	Числ.мет лекция,241(1) Крыжановская Т.В.	Числ.мет лекция,241(1) Крыжановская Т.В.
Ср-3		ВишМат_ММДО лаб.раб,318(1) Буяджи В.И.		ВишМат_ММДО лаб.раб,318(1) Буяджи В.И.
Ср-4				
Ср-5				
Чт-1	Електротехніка лекция,242(1) Горьев С.А.	Електротехніка лекция,242(1) Горьев С.А.	Електротехніка лекция,242(1) Горьев С.А.	Електротехніка лекция,242(1) Горьев С.А.
Чт-2	ин.яз	ин.яз	ин.яз	ин.яз
Чт-3	физкультура	физкультура	физкультура	физкультура
Чт-4				
Чт-5				
Пт-1	ВишМат_ММДО лаб.раб,318(1) Буяджи В.И.	Електротехніка лаб.раб,126(1) Горьев С.А.	ВишМат_ММДО лаб.раб,318(1) Буяджи В.И.	Електротехніка лаб.раб,126(1) Горьев С.А.
Пт-2	Числ.мет лаб.раб,324(1) Крыжановская Т.В.	Числ.мет лаб.раб,327б(1) Шпинарева И.М.	Числ.мет лаб.раб,324(1) Крыжановская Т.В.	Числ.мет лаб.раб,327б(1) Шпинарева И.М.
Пт-3	Електротехніка лаб.раб,126(1) Горьев С.А.	Фізика лаб.раб,512(2) Петров П.П.	Електротехніка лаб.раб,126(1) Горьев С.А.	
Пт-4				
Пт-5				
Сб-1				
Сб-2				
Сб-3				
Сб-4				
Сб-5				

Рисунок 9.2 – Вигляд розкладу занять групи

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

Розробка програми автоматизованого складання розкладу занять є актуальним завданням, оскільки не існує подібних програм з відкритим кодом. Програми, що пропонуються для продажу коштують досить дорого, якщо вони виконують всі вимоги до гарної програми складання розкладу занять у закладі вищої освіти.

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 потокових лекцій та занять, що проводяться з поділом групи на підгрупи.

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.

Результатами виконання даної частини магістерської роботи є:

- розроблена модель обліку у програмі складання розкладу занять потокових лекцій на лабораторних занять з поділом на підгрупи;
- відповідно до моделі реструктурована база даних;
- внесені зміни у алгоритм сортування списку занять для урахування потокових лекцій;
- розроблений алгоритм обліку у програмі складання розкладу занять потокових лекцій та лабораторних занять з поділом на підгрупи;
- розроблено серверне програмне забезпечення програми автоматизованого складання розкладу занять;
- Розроблене серверне програмне забезпечення включає в себе 8 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 від користувача введення великих масивів вихідних даних.

Дані по заняттям академічних груп автоматично розраховуються у базі даних після введення у базу даних інформації по навчальним планам факультетів по прикріплення кожної групи до плану. Дані по прикріпленню

викладчів до занять заносяться автоматично при виборі кафедрами викладача для кожного заняття кафедри. Так саме автоматично у базі даних ставляться відмітки про об’єднання груп у потік та розділення групи на підгрупи при виборі відповідних дій у програмі «АРМ працівника кафедри».

ПЕРЕЛІК ПОСИЛАНЬ

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

2 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. – № 1 (15) – С.43-50.

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 «Розрахунок розкладу»

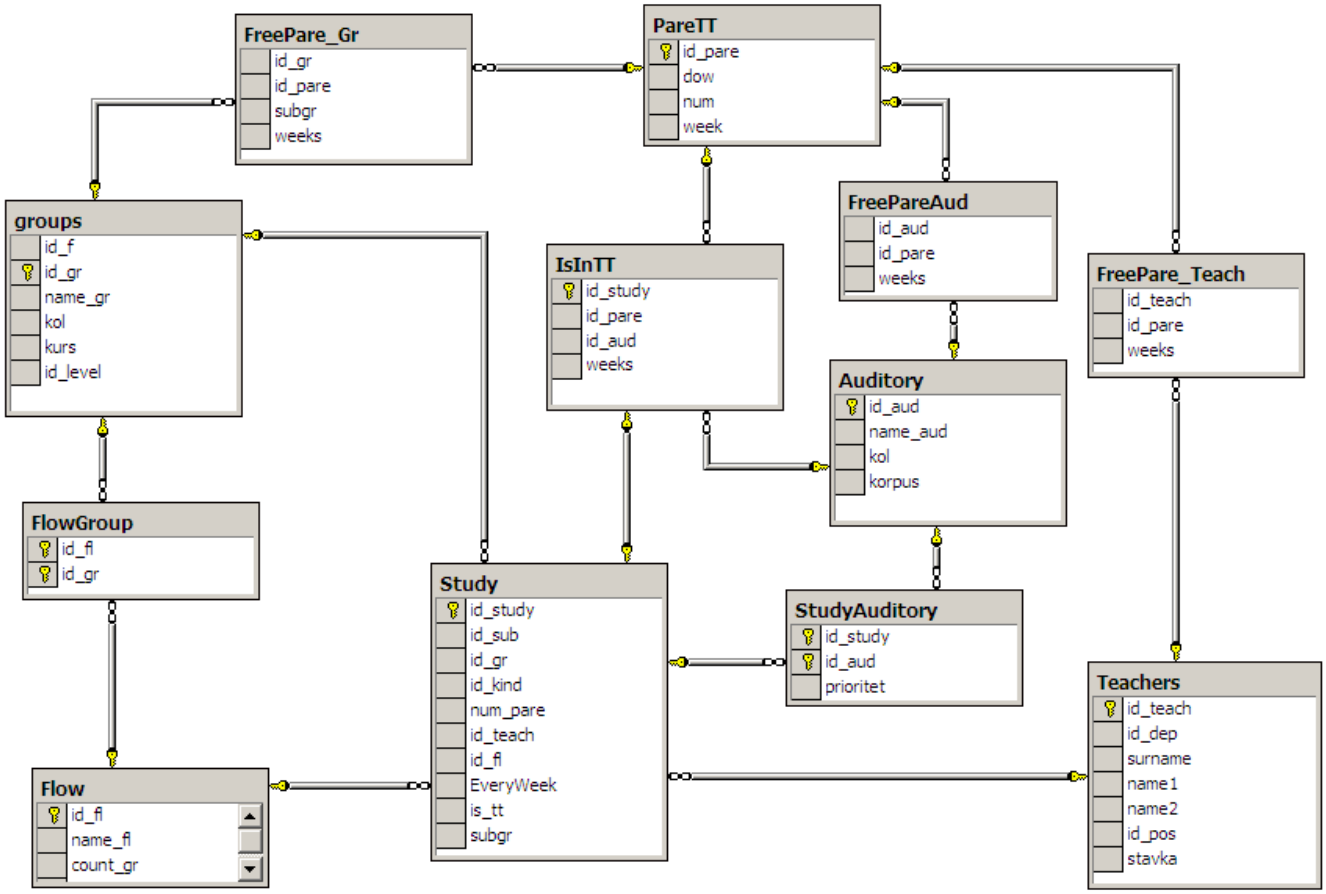

Рисунок А.1 – Логічна схема БД «Розклад занять» – підсхема «Вихідні дані»

ДОДАТОК Б

Вихідний код серверного ПЗ

```
CREATE proc FillFlowParePriorWeek
@id_study int,
@id_fl smallint,
@week bit
as
begin

      declare @@k smallint, @@k1 smallint, @@k2 smallint, @@val numeric(7,2)
      declare @@id_sub smallint,@@num_pare tinyint, @@id_kind tinyint, @@id_gr
smallint
      delete GrParePrior
      delete GrParePriorWeek

      DECLARE GroupStudy_cursor CURSOR FAST_FORWARD FOR
      SELECT id_sub,num_pare, id_kind
      From Study Where id_study=@id_study
      OPEN GroupStudy_cursor
      FETCH NEXT FROM GroupStudy_cursor INTO @@id_sub,@@num_pare, @@id_kind
      CLOSE GroupStudy_cursor
      DEALLOCATE GroupStudy_cursor

      DECLARE FlowGroup_cursor CURSOR FAST_FORWARD FOR
      SELECT id_gr FROM  FlowGroup
      WHERE id_fl=@id_fl
      OPEN FlowGroup_cursor
      FETCH NEXT FROM FlowGroup_cursor INTO @@id_gr
      WHILE @@FETCH_STATUS = 0
      BEGIN
            set @id_study =(select id_study from Study where id_gr=@@id_gr
                  and id_sub=@@id_sub and num_pare=@@num_pare and
id_kind=@@id_kind)

            exec FillGrParePriorWeek @@id_gr, @id_study, @week

            FETCH NEXT FROM FlowGroup_cursor INTO @@id_gr
      END
      CLOSE FlowGroup_cursor
      DEALLOCATE FlowGroup_cursor


      delete FlowPrior
      if @week=1
      begin
            delete GrParePrior where prioritet=0 or prioritet is null

            insert into FlowPrior
            select id_pare,weeks=null, Pmax=MAX(prioritet),
Psum=SUM(prioritet),Pmin=MIN(prioritet), k=count(*)
            from GrParePrior
            Group by id_pare

            delete GrParePriorWeek where prioritet=0 or prioritet is null

            insert into FlowPrior
            select id_pare,weeks=null, Pmax=MAX(prioritet),
Psum=SUM(prioritet),Pmin=MIN(prioritet), k=count(*)
            from GrParePriorWeek
```

```
            where weeks is null
            Group by id_pare

      end
      else
      begin
            delete GrParePriorWeek where prioritet=0 or prioritet is null

            insert into FlowPrior
            select id_pare,weeks, Pmax=MAX(prioritet),
Psum=SUM(prioritet),Pmin=MIN(prioritet), k=count(*)
            from GrParePriorWeek
            where weeks is not null
            Group by id_pare,weeks
      end

      set @@k=(select k=max(k) from FlowPrior)
      delete FlowPrior where k<@@k

      set @@val=(select value_c from Coeff where name_c= 'PriorSr')
      set @@k1=(select k=count(*) from FlowPrior)
      SET @@k2=(select k=count(*) from FlowPrior where Pmin>=@@val)

      if @@k2>@@k1/3
            delete FlowPrior where Pmin<@@val
      else
      begin
            SET @@k2=(select k=count(*) from FlowPrior where Pmin>=@@val/2)
            if @@k2>@@k1/3
                  delete FlowPrior where Pmin<@@val/2
      end
end
GO


CREATE  proc Study2TTEveryWeekFlow
@id_study int,
@id_gr smallint,
@id_teach smallint,
@id_fl 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

      DECLARE @@id_sub smallint, @@num_pare tinyint, @@id_kind tinyint
      DECLARE GroupStudy_cursor CURSOR FAST_FORWARD FOR
      SELECT id_sub,num_pare, id_kind
      From Study Where id_study=@id_study
      OPEN GroupStudy_cursor
      FETCH NEXT FROM GroupStudy_cursor INTO @@id_sub,@@num_pare, @@id_kind
      CLOSE GroupStudy_cursor
      DEALLOCATE GroupStudy_cursor

      DECLARE FlowGroup_cursor CURSOR FAST_FORWARD FOR
      SELECT id_gr FROM  FlowGroup
      WHERE id_fl=@id_fl
      DECLARE @@id_gr smallint, @@id_study int
      OPEN FlowGroup_cursor
```

```
      FETCH NEXT FROM FlowGroup_cursor INTO @@id_gr
      WHILE @@FETCH_STATUS = 0
      BEGIN
            set @@id_study =(select id_study from Study where id_gr=@@id_gr
                        and id_sub=@@id_sub and num_pare=@@num_pare
                        and id_kind=@@id_kind)
            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

            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

            FETCH NEXT FROM FlowGroup_cursor INTO @@id_gr
      END
      CLOSE FlowGroup_cursor
      DEALLOCATE FlowGroup_cursor
END
GO


CREATE  proc Study2TTWeekFlow
@id_study int,
@id_gr smallint,
@id_teach smallint,
@id_fl smallint,
@id_pare smallint,
@id_aud smallint,
@weeks bit
as
BEGIN
      DECLARE @@id_sub smallint, @@num_pare tinyint, @@id_kind tinyint
      DECLARE GroupStudy_cursor CURSOR FAST_FORWARD FOR
      SELECT id_sub,num_pare, id_kind
      From Study Where id_study=@id_study
      OPEN GroupStudy_cursor
      FETCH NEXT FROM GroupStudy_cursor INTO @@id_sub,@@num_pare, @@id_kind
      CLOSE GroupStudy_cursor
      DEALLOCATE GroupStudy_cursor

      DECLARE FlowGroup_cursor CURSOR FAST_FORWARD FOR
      SELECT id_gr FROM  FlowGroup
      WHERE id_fl=@id_fl
      DECLARE @@id_gr smallint, @@id_study int
      OPEN FlowGroup_cursor
      FETCH NEXT FROM FlowGroup_cursor INTO @@id_gr
      WHILE @@FETCH_STATUS = 0
      BEGIN
            set @@id_study =(select id_study from Study where id_gr=@@id_gr
                              and id_sub=@@id_sub and num_pare=@@num_pare
                              and id_kind=@@id_kind)
            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_study=@@id_study
            delete StudyFull where id_gr=@@id_gr 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)

            FETCH NEXT FROM FlowGroup_cursor INTO @@id_gr
```

```
      END
      CLOSE FlowGroup_cursor
      DEALLOCATE FlowGroup_cursor
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
```