

Циклова комісія комп'ютерної інженерії

НАВЧАЛЬНІ МАТЕРІАЛИ

до самостійного вивчення тем з дисципліни

ОРГАНІЗАЦІЯ БАЗ ДАНИХ

спеціальність	123 Комп'ютерна інженерія
освітня програма	Обслуговування комп'ютерних систем і мереж (назва освітньої програми)
відділення	Комп'ютерних технологій та систем (назва відділення)

**Дніпро
2023 рік**

ЗМІСТ

1.	Класифікація сутностей-об'єктів в концептуальній моделі даних. Приклади проєктування реляційної бази дани	3
2.	Цілісність даних. Цілісність бази даних	20
3.	Тригери: особливості використання	32
4.	Розподілені бази даних	43

КРИТЕРІЇ ОЦІНЮВАННЯ роботи студента над темою для самостійного вивчення

В якості результату самостійної роботи над темою студент повинен надати конспект теми та відповіді на контрольні питання, надані в плані самостійного вивчення теми.

Розподіл балів та критерії оцінювання виконаного завдання наведено в таблиці

Опрацювання теми для самостійного вивчення		
Бали	Критерій	
2 Конспект теми	2	Конспект теми повний, логічно-послідовний. Студент зумів виділити та опрацювати основні положення теми, виділені в плані самостійного вивчення теми.
	1,5	Конспект повний, але при цьому допущені невеликі пропуски, що не спотворили логічного і інформаційного змісту теми.
	1	Конспект неповно або непослідовно розкриває зміст матеріалу теми, але в конспекті виділені основні положення теми.
	0,5	Конспект не розкриває основний зміст навчального матеріалу, або студент не зумів виділити та опрацювати основні положення теми.
2 Відповіді на контрольні питання	2	Студент правильно та повному об'ємі відповідає на всі контрольні питання, надані в плані самостійного вивчення теми.
	1,5	Студент правильно та повному об'ємі відповідає на 70% контрольних питань, наданих в плані самостійного вивчення теми.
	1	Студент правильно та повному об'ємі відповідає на 50% контрольних питань, наданих в плані самостійного вивчення теми.
	0,5	Студент правильно та повному об'ємі відповідає не більше ніж на 30% контрольних питань, наданих в плані самостійного вивчення теми.
Максимальна кількість балів: 4		

План самостійного вивчення теми № 1

з навчальної дисципліни Організація баз даних

Тема: Класифікація сутностей-об'єктів в концептуальній моделі даних. Приклади проектування реляційної бази даних

Мета: Розібрати приклад моделювання локальної предметної області (Про). Освоїти поняття спеціалізації, генералізації та категоризації; суперкласу, складеного об'єкту.

знати: зміст понять:

- об'єкт;
- атрибут;
- бінарний зв'язок;
- зв'язок "один-до-багатьох";
- потужність;
- показник кардинальності;
- ступінь участі;
- ключ;
- рекурсивний зв'язок;
- складений об'єкт.

вміти:

- вміти будувати концептуальну модель деякої предметної області, використовуючи поняття складеного об'єкта..

Час – 5 годин

Навчальні питання:

1 Приклад моделювання локальної предметної області (Про).

2 Складені об'єкти

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

По-перше

Суперклас - це об'єкт, що включає різні підкласи, які необхідно представити в моделі даних. Підклас - це об'єкт, що є членом суперкласу, але виконує окрему роль у ньому. Суперклас може мати кілька різних підкласів.

- По-друге* Спеціалізація являє собою процес збільшення розходжень між окремими екземплярами об'єкта за з виділення їхніх відмітних характеристик. Цей процес являє собою спадний підхід до визначення множини суперкласів і пов'язаних з ним підкласів.
- По-третє* Генералізація являє собою висхідний підхід, а саме: процес відомості розходжень між об'єктами до мінімуму шляхом виділення їхніх загальних характеристик. Цей підхід дозволяє створити суперклас на основі різних вихідних підкласів.
- По-четверте* Підклас буде категорією, якщо він зв'язаний відразу з декількома суперкласами різних типів. Підклас категорії має вибіркове спадкування, що означає спадкування кожним екземпляром категорії атрибутів тільки одного суперкласу. Категорія може бути також двох типів: з повною участю й частковою участю.

Література

1. Доценко С. І. Організація та системи керування базами даних: Навч. посібник. [Текст] – Харків: УкрДУЗТ, 2023. – 117 с., рис. 92, табл. 3.
2. Трофименко О. Г. Організація баз даних :Навч. посібник / О. Г. Трофименко, Ю. В. Прокоп, Н. І. Логінова, І. М. Копитчук. 2-ге вид. виправ. і доповн. [Текст] – Одеса : Фенікс, 2019. – 246 с.
3. Харів Н. О. Бази даних та інформаційні системи: навчальний посібник / Н. О. Харів. [Текст] – Рівне : НУВГП, 2018. – 127 с.
4. Ярцев В.П. Організація баз даних та знань: навчальний посібник./ В.П. Ярцев [Текст] – К. ДУТ 2018.-214с.

Навчальні матеріали до самостійного вивчення теми

1 Приклад моделювання локальної предметної області (Про).

За допомогою розглянутих вище понять можуть бути отримані ER-моделі для більшості схем баз даних у традиційних адміністративно-управлінських додатках. Якщо Про велика, то побудова її концептуальної моделі буде протікати більш успішно, якщо Про розбити на трохи локальних предметних областей. Обсяг локальної Про вибирається таким чином, щоб у неї входило не більше 6 - 7 об'єктів. Відправними елементами для побудови ER-моделі локальної Про дуже часто є використовувані в організації документи.

Припустимо, що визначено локальна Про: поставка товарів на склад. Нехай форма поставки має вигляд, як на рисунку 1.

Постачальник			
Адреса постачальника			
Індекс постачальника			
Поставка			
Дата поставки	Індекс поставки	№ складу	
Товар			
Індекс	Найменування	Ціна	Кількість

Рисунок 1. Форма поставки

Використовуючи наведену форму, можна побудувати концептуальну модель цієї невеликої локальної предметної області. Аналізована форма містить наступну інформацію: **Постачальник**, **Індекс постачальника**, **Адреса постачальника**, **Товар**, **Індекс товару**, **Ціна товару**, **Кількість товару**, **Поставка**, **Індекс поставки**, **Дата поставки** й **Номер складу**.

Виділимо два об'єкти: **ПОСТАЧАЛЬНИК** і **ТОВАР** (рис. 2). Атрибути, що залишилися, характеризують об'єкт - **ПОСТАВКА**. Сформуємо його й установимо певні

типи бінарних зв'язків між трьома об'єктами, виходячи з наступних міркувань: той самий постачальник може здійснити ряд поставок, але кожна поставка здійснюється тільки одним постачальником.



Рисунок 2. ПрО Поставка

Потужність зв'язку між об'єктами ПОСТАВКА й ТОВАР повинна бути встановлена М:N, тому що кожна поставка може містити кілька товарів, і той самий товар може втримуватися в декількох поставках. Виходячи з вищесказаного, діаграма моделі предметної області ПОСТАВКА прийме такий вид, як на рисунку 3. Атрибути **Індекс постачальника**, **Індекс поставки** й **Індекс товару** були введені для однозначної ідентифікації екземплярів розглянутих об'єктів, тому що жоден з інших атрибутів не підходить на цю роль. Такі ідентифікаційні атрибути називаються первинними ключами. Для роботи проектованої системи може знадобитися також виділення ключів іншого класу. Наприклад, таких, що кожному значенню подібного ключа може задовольняти деяку безліч елементів об'єктної безлічі, а не один. Такі ключі називаються вторинними ключами. При побудові концептуальної моделі варто уникати надмірності інформації. Після того, як виділені об'єкти, ключі, визначають і видаляють наявні надлишкові зв'язки. Велика увага приділяється аналізу атрибутів. У добре спроектованій БД повинне дотримуватися правило: *серед атрибутів об'єкта повинна спостерігатися залежність описового атрибута від ключового, але не повинна існувати залежність між описовими атрибутами.*

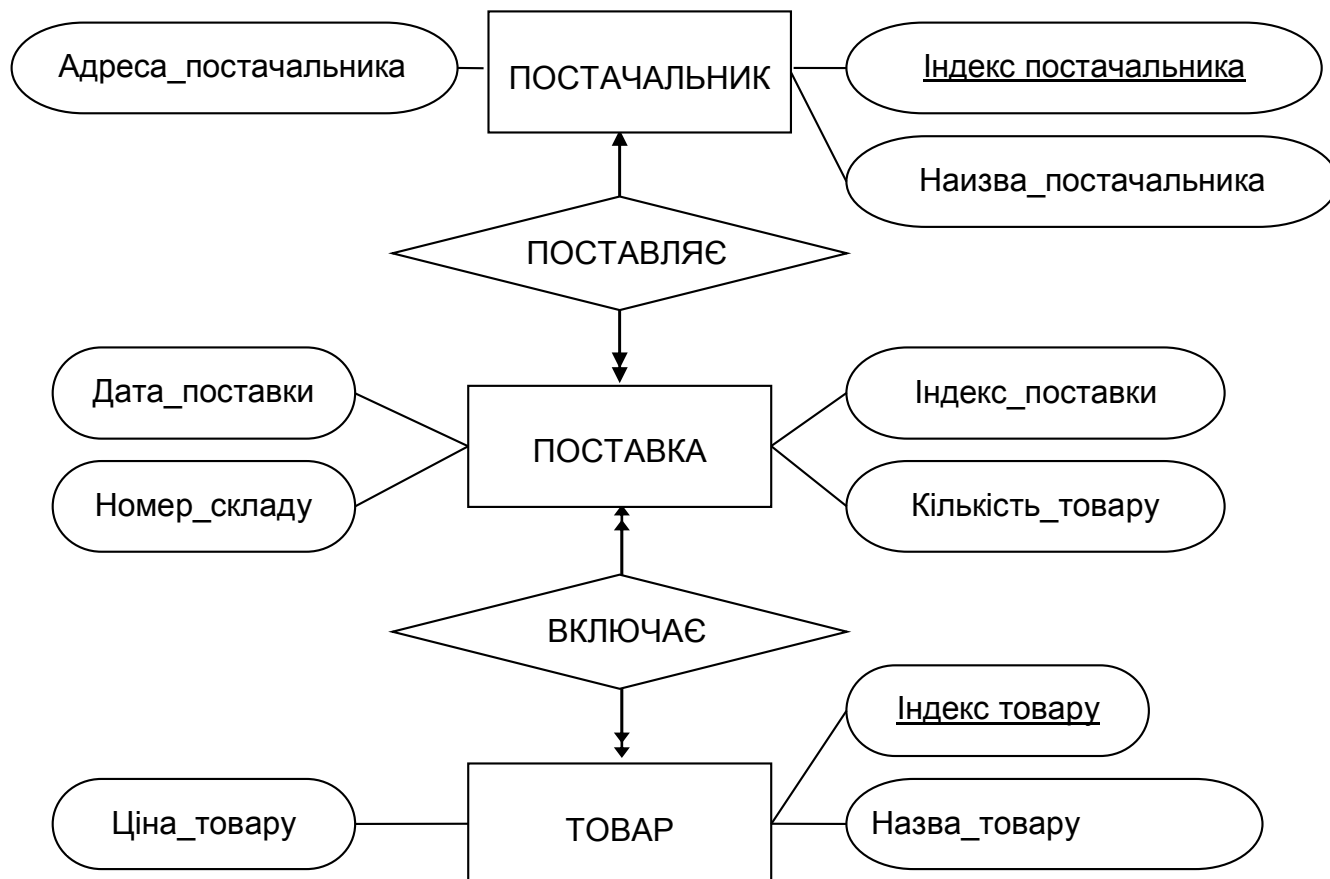


Рисунок 3. Діаграма моделі предметної області ПОСТАВКА

Завершальним етапом побудови концептуальної моделі досліджуваної Про є специфікація всіх об'єктів, що входять у модель. Для даного прикладу результати цього кроку повинні бути зведені до наступного:

- Специфікація об'єктів:

ПОСТАЧАЛЬНИК:	<u>Індекс постачальника</u>	– ідентифікаційний атрибут
	Адреса_постачальника	– описовий атрибут
	Назва_постачальника	– описовий атрибут
ПОСТАВКА:	<u>Індекс поставки</u>	– ідентифікаційний атрибут
	Кількість_товару	– описовий атрибут
	Дата_поставки	– описовий атрибут
	Номер_складу	– описовий атрибут
ТОВАР:	<u>Індекс товару</u>	– ідентифікаційний атрибут
	Назва_товару	– описовий атрибут
	Ціна товару	– описовий атрибут

- Специфікація типів зв'язків:

Поставляє:	зв'язок	ПОСТАЧАЛЬНИК	↔	ПОСТАВКА	1:N
Включає:	зв'язок	ПОСТАВКА	↔	ТОВАР	M:N

- Специфікація атрибутів:

Індекс постачальника: символний, 6 символів
Адреса_постачальника: символний, 50 символів

Ціна_товару: гроршовий

Сформовані специфікації заносяться в словник даних. Після створення моделей кожної виділеної Про виробляється об'єднання локальних концептуальних моделей в одну загальну.

Спеціалізація й генералізація. Для задоволення нових вимог, висунутих усе більше складними додатками, у семантичне моделювання були уведені додаткові концепції, що розширюють його можливості. Така модель одержала назву розширеної ER-моделі (EER-модель). Вона включає всі концепції ER-моделі плюс концепції спеціалізації, генералізації й категоризації. Всі ці нововведення створюють додаткові можливості для моделювання неоднорідних по своїй структурі об'єктів. Додаткові концепції базуються на таких поняттях, як суперклас і підклас, а також використовують процес спадкування атрибутів.

Суперклас - це об'єкт, що включає різні підкласи, які необхідно представити в моделі даних. Підклас - це об'єкт, що є членом суперкласу, але виконує окрему роль у ньому. Суперклас може мати кілька різних підкласів. Так, наприклад, підкласи АСИС-ТЕНТ, СТАРШИЙ ВИКЛАДАЧ, ДОЦЕНТ, ПРОФЕСОР є членами суперкласу ВИКЛАДАЧ. Це означає, що кожен екземпляр підкласу є в той же час й екземпляром суперкласу. Зв'язок між суперкласом і підкласом ставиться до типу "один до одному".

Використання понять суперкласу й підкласів дозволяє при моделюванні виділити для підкласу свої власні атрибути й атрибути, наслідувані їм від суперкласу. Так, наприклад, підклас ДОЦЕНТ повинен мати ті ж атрибути, що й всі викладачі - це наслідувані їм від суперкласу ВИКЛАДАЧ атрибути. Однак він має й свій власний атрибут, що не

визначається для інших категорій викладачів - Номер_диплома_доцента. Під час відсутності виділених підкласів у цій ситуації довелося б для об'єкта ВИКЛАДАЧ увести такий атрибут, який би мав невизначене значення для всіх викладачів, що не є доцентами. Підклас може мати свої власні зв'язки, які не підходять для всіх екземплярів суперкласу. Наприклад, професорові дозволяється керувати аспірантами.

На діаграмі (рис. 4) підкласи з'єднуються лініями із кружком, що у свою чергу з'єднується із суперкласом. На кожній лінії, що йде від підкласу, розташовується U-образний символ, що позначає напрямок включення. Верхня частина U "відкривається" у бік суперкласу. У середині кружка розташовується буква D, якщо підкласи не перетинаються, і буква ПРО - для пересічних підкласів. В останньому випадку екземпляр суперкласу може бути членом відразу декількох підкласів. Зображена на діаграмі ситуація виключає перетинання підкласів, тому в кружок поміщений символ D.

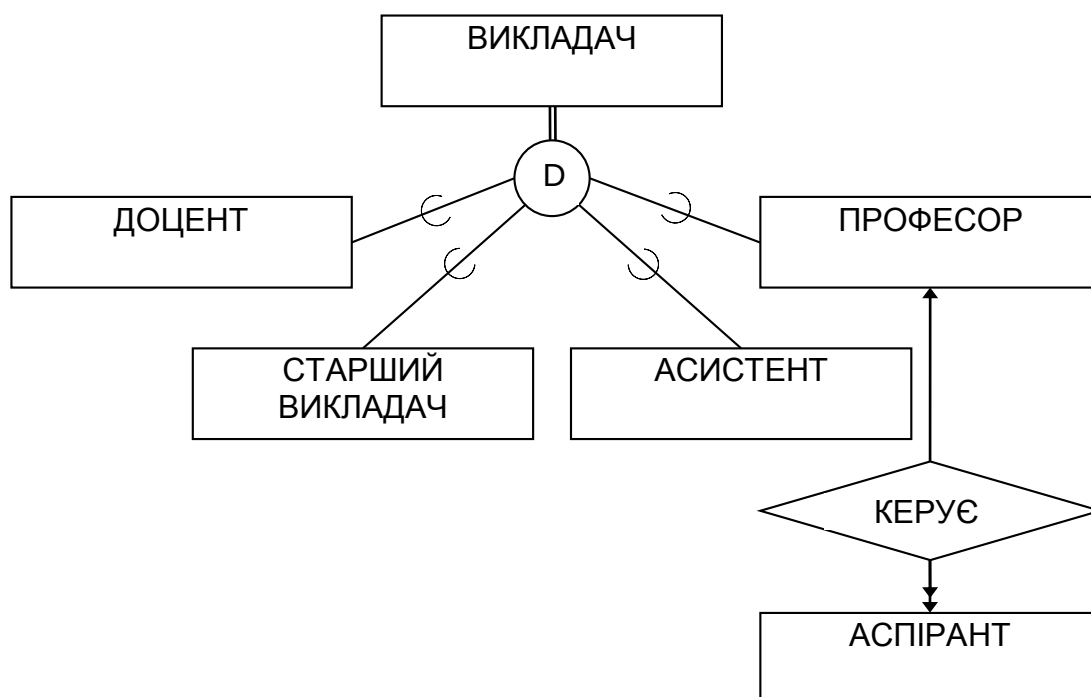


Рисунок 4. Діаграма з використанням понять суперклас і підклас

Введення понять суперкласів і підкласів, дозволяючи уникнути повторного опису атрибутів, заощаджує час проектувальника, підвищує читабельність, скорочує обсяг невизначених значень. Крім цього використання цих понять збільшує кількість семантичної інформації, що втримується в моделі, дозволяє неї зробити більше зрозумілої. В EER-моделі розрізняють два процеси, що мають протилежні напрямки: це спеціалізація й генералізація.

Спеціалізація являє собою процес збільшення розходжень між окремими екземплярами об'єкта за з виділення їхніх відмітних характеристик. Цей процес являє собою спадний підхід до визначення безлічі суперкласів і пов'язаних з ним підкласів. При виявленні набору підкласів об'єкта виконується також виділення специфічних для кожного підкласу атрибутів й їхніх зв'язків з іншими об'єктами. Специфічні для підкласу атрибути з'єднуються з ним безпосередньо. Опираючись на різні характеристики об'єкта, для того самого об'єкта можна виділити трохи незалежних спеціалізацій. Наприклад, той же об'єкт ВИКЛАДАЧ можна специфіцизувати, розбивши його на два підкласи: ОСНОВНА_РОБОТА й РОБОТА_ПО_СУМІСНИЦТВУ. До першого підкласу ставляться ті викладачі, для яких ця робота є основною, а до другого - ті, які неї сполучають із іншою основною роботою. Це приклад спеціалізації на пересічні підкласи, оскільки той самий викладач може бути учасником обох підкласів. Така ситуація, наприклад, виникає при наявності вакансій, тобто тоді, коли кількість викладачів менше необхідного. У цій ситуації деякому викладачеві може бути доручене виконання додаткового навантаження, а він сам буде оформлений ще і як сумісник.

Генералізація являє собою висхідний підхід, а саме: процес відомості розходжень між об'єктами до мінімуму шляхом виділення їхніх загальних характеристик. Цей підхід дозволяє створити суперклас на основі різних вихідних підкласів. Допустимо, що в наявності є наступні об'єкти: АСИСТЕНТ, СТАРШИЙ ВИКЛАДАЧ, ДОЦЕНТ, ПРОФЕСОР. Застосування методу генералізації до них приведе до виявлення їхніх загальних властивостей і зв'язків. Спільність їх полягає в тім, що всі вони є викладачами, а виходить, можна визначити суперклас ВИКЛАДАЧ, що буде мати всі спільно використовувані атрибути.

На процедуру спеціалізації, крім обмеження перетинання, може бути накладене ще одне обмеження, що називається обмеженням участі, і воно може бути повним або частковим. При спеціалізації з повною участю кожен екземпляр суперкласу повинен бути екземпляром цієї спеціалізації, тобто належати якому-небудь виділеному підкласу. Для позначення на діаграмах повної участі між суперкласом і кружком проводять подвійну лінію. Прикладом спеціалізації з повною участю можуть служити суперклас ВИКЛАДАЧ і його підкласи, де кожен викладач повинен ставитися до однієї із зазначених чотирьох категорій (рис. 4). Спеціалізація із частковою участю означає, що кожен

екземпляр об'єкта не обов'язково повинен бути членом підкласу цієї спеціалізації. Для позначення на діаграмах часткової участі між суперкласом і кружком проводять одинарну лінію. Підкласи успадковують не тільки атрибути, але й всі зв'язки суперкласу (рис. 5).

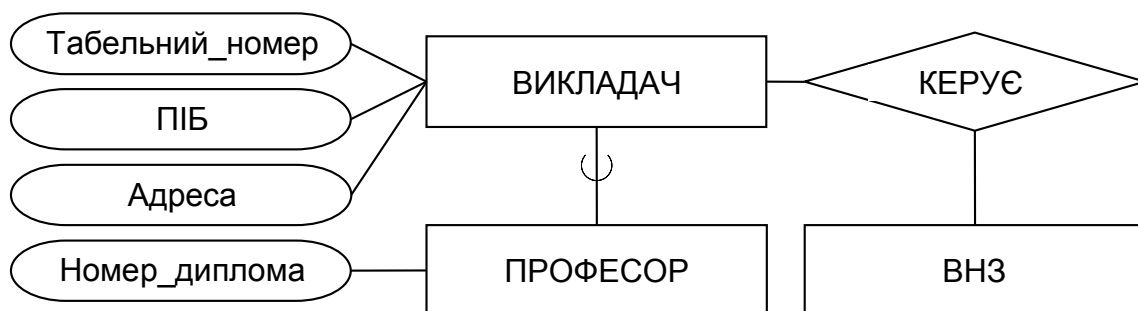
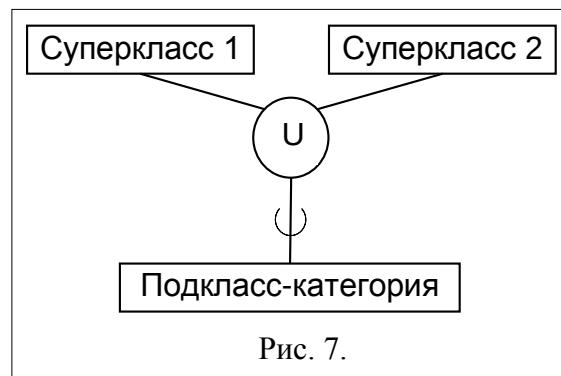
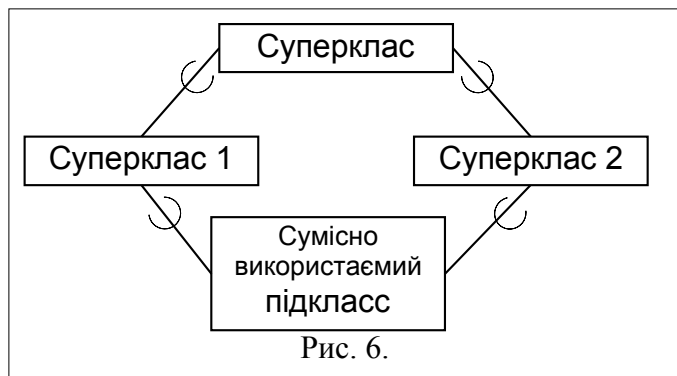


Рисунок 5. Спадкування підкласом зв'язків суперкласу

Категоризація. Ієрархія об'єктів розроблювальної моделі може бути досить складної. Так, наприклад, можна представити таку ситуацію, при якій один підклас може бути пов'язаний з декількома суперкласами, які у свою чергу є підкласами одного загального для них суперкласу. У цьому випадку даний підклас називається *спільно використовуваним підкласом*, так що кожен екземпляр підкласу повинен одночасно бути екземпляром кожного пов'язаного з ним суперкласу (рис. 6). Підклас буде категорією, якщо він зв'язаний відразу з декількома суперкласами різних типів (рис. 7). Підклас категорії має вибіркове спадкування, що означає спадкування кожним екземпляром категорії атрибутів тільки одного суперкласу. Категорія може бути також двох типів: з повною участю й частковою участю. Найбільш твердим є обмеження, пов'язане з повною участю, оскільки воно припускає, що кожен екземпляр всіх суперкласів повинен бути представлений у даній категорії. При частковій же участі присутність у категорії всіх екземплярів всіх суперкласів необов'язково. При повній участі від підкласу-категорії йде подвійна лінія до кружка категоризації, при частковій участі - одинарна.



Приклад. Розглянемо приклад створення EER-моделі предметної області НАВЧАЛЬНИЙ ПРОЦЕС. Виділені в ній об'єкти й зв'язки не охоплюють всі нюанси даної проблеми (постановка завдання не претендує на повноту), але остаточна діаграма цієї моделі вийде досить складної. Для полегшення сприйняття складних діаграм дозволяється не наносити атрибути об'єктів (або позначати тільки ідентифікаційні атрибути), а зв'язку між об'єктами тільки надписувати, опускаючи їхнього позначення. У результаті збору вимог у даній Про і їхній аналіз були сформульовані наступні специфікації, які наведені в таблиці 1.

Таблиця 1. Об'єкти і їхні атрибути

ОБ'ЄКТ	АТРИБУТ	ПЕРВИННИЙ КЛЮЧ
ФАКУЛЬТЕТ	Код факультету	Код факультета
	Назва факультету	
КАФЕДРА	Код кафедри	Код кафедри
(загальна, випускна)	Назва кафедри	
СОТРУДНИК	Табельний номер	Табельний номер
(інж_лабор_склад	ПІБ	
викладач	Стаж	
лектор	Посада	
асистент	Педагогічний стаж	
старший викладач)		
ДОЦЕНТ	Диплом доцента	
ПРОФЕСОР	Диплом професора	
СПЕЦІАЛЬНІСТЬ	Шифр спеціальності	Шифр спеціальности
	Назва спеціальності	
КУРС	Номер курсу	Номер курсу
ДИСЦИПЛІНА	Код дисципліни	Код дисципліни
	Назва дисципліни	
ГРУППА	Код групи	Код групи
	Староста	
	Кількість студентів	
СТУДЕНТ	Номер залікової книжки	Номер залікової книжки
	ПІБ	
	Адреса	
ДИПЛОМНИК	Тема диплому	

– Факультет поєднує ряд кафедр. Серед кафедр є загальноосвітні кафедри й кафедри, що випускають фахівців. Кафедра може готувати фахівців з декількох спеціальностей. Особи, що працюють на кафедрі, ставляться до категорії СПІВРОБІТНИК.

– Всі співробітники діляться на дві частини: обслуговуючий персонал - інженерно-лаборантський склад і викладацький склад. Кожен викладач повинен ставитися до однієї із чотирьох категорій: асистент, старший викладач, доцент, професор, причому всі категорії, крім категорії асистент, входять у лекторський склад. Представникам лекторського складу поручається читання лекцій і керівництво дипломним проектуванням.

– Викладання студентам кожної спеціальності ведеться відповідно до затвердженого навчального плану, де визначені дисципліни, що підлягають вивченню, види занять, що відводять під них кількість годин, а також визначена форма контролю знань по цій дисципліні: залік або іспит.

– По даній спеціальності навчання студентів ведеться на одному або більше курсів. На кожному курсі може бути одна або кілька груп однієї й тієї ж спеціальності, де навчається ряд студентів.

Для спрощення концептуальної моделі бази даних залишилися не проробленими такі моменти навчального процесу, як: хто й де веде певний вид занять. Із процесу навчання по даній дисципліні виділені тільки лекції, а всі інші види занять, а також форми контролю віднесені до об'єкта ІНШІ.

Об'єкти, виділені в розглянутій моделі, а також стосовні до них атрибути, які виявлені зі специфікацій вимог користувачів, наведені в таблиці. Вивчення складу атрибутів кожного об'єкта дозволило виділити ті з них, які можна використати як первинні ключі. Вони також поміщені в таблицю 1. Неважко помітити, що не всі позиції в колонку Первинний ключ заповнені. Вільними залишилися ті позиції, які ставляться до об'єктів-підкласів, отриманим шляхом спеціалізації. Оскільки підклас успадковує всі атрибути суперкласу, а отже, і ідентифікаційні атрибути й кожен екземпляр підкласу зв'язаний тільки з одним екземпляром суперкласу, той й ідентифікаційний атрибути в них ті самі. Для виділених об'єктів необхідно встановити найважливіші типи зв'язків, що відповідають тим, що існують у реальному світі. Для кожного зв'язку варто також визначити показник кардинальності й ступінь участі сторін у зв'язку. Всі зв'язки і їхні з наведені в таблиці 2.

Таблиця 2. Типи зв'язків та їх характеристики

Зв'язок	Об'єкти	Показник кардинальності	Степень участі
ВХОДИТЬ	ФАКУЛЬТЕТ КАФЕДРА	1:N	Повна Повна
ПРАЦЮЄ	КАФЕДРА СПІВРОБІТНИК	1:N	Повна Часткова
ГОТУЄ	ВИПУСКНА КАФЕДРА СПЕЦІАЛЬНІСТЬ	1:N	Повна Повна
ЗАТВЕРДЖЕНИЙ	СПЕЦІАЛЬНІСТЬ НАВЧАЛЬНИЙ_ПЛАН	1:1	Повна Повна
МАЄ	СПЕЦИАЛЬНОСТЬ КУРС	1:N	Повна Повна
СКЛАДАЄТЬСЯ	КУРС ГРУПА	1:N	Повна Повна
ВИКЛАДАЄ	ВИКЛАДАЧ ДИСЦИПЛІНА	N:M	Повна Повна
ВЧИТЬСЯ_В	ГРУППА СТУДЕНТ	1:N	Повна Повна
ВИВЧАЄ	ДИСЦИПЛІНА СТУДЕНТ	N:M	Повна Повна
КЕРУЄ_ДИПЛОМНИКОМ	ЛЕКТОР ДИПЛОМНИК	1:N	Часткова Повна
ГОТУЄ_ДО_ЗАХИСТУ	СПЕЦІАЛЬНІСТЬ ДИПЛОМНИК	1:N	Повна Часткова

Наведена в таблиці інформація повинна бути використана при побудові ER-діаграм (рис. 8).

Оскільки для спрощення діаграми символи для позначення зв'язків не використалися, то замість них на лініях зв'язку поміщені крапки, що розділяють лінії зв'язку на дві частини. Кожна така частина використовується для позначення ступеня участі об'єкта, що прилягає з відповідної сторони до крапки у зв'язку. Якщо об'єкт приєднується до крапки подвійною лінією, то ступінь його участі у зв'язку - повна, якщо ж одинарної, то ступінь його участі у зв'язку - часткова.

Для більшості об'єктів розглянутої предметної області визначений повний ступінь участі у зв'язку. Це значить, що кожен екземпляр об'єкта повинен брати участь у зв'язку. Виключення з даного правила становлять наступні об'єкти: СПІВРОБІТНИК у зв'язку ПРАЦЮЄ (не кожен співробітник працює на кафедрі, оскільки у вузі існують й інші

структури), ЛЕКТОР у зв'язках ЧИТАЄ й КЕРУЄ_ДИПЛОМНИКОМ (він може й не читати лекції, і не керувати дипломним проектуванням), СПЕЦІАЛЬНІСТЬ у зв'язку ГОТОВИТЬ_ДО_ЗАХИСТУ (можливо, по даній спеціальності ще немає випускників).

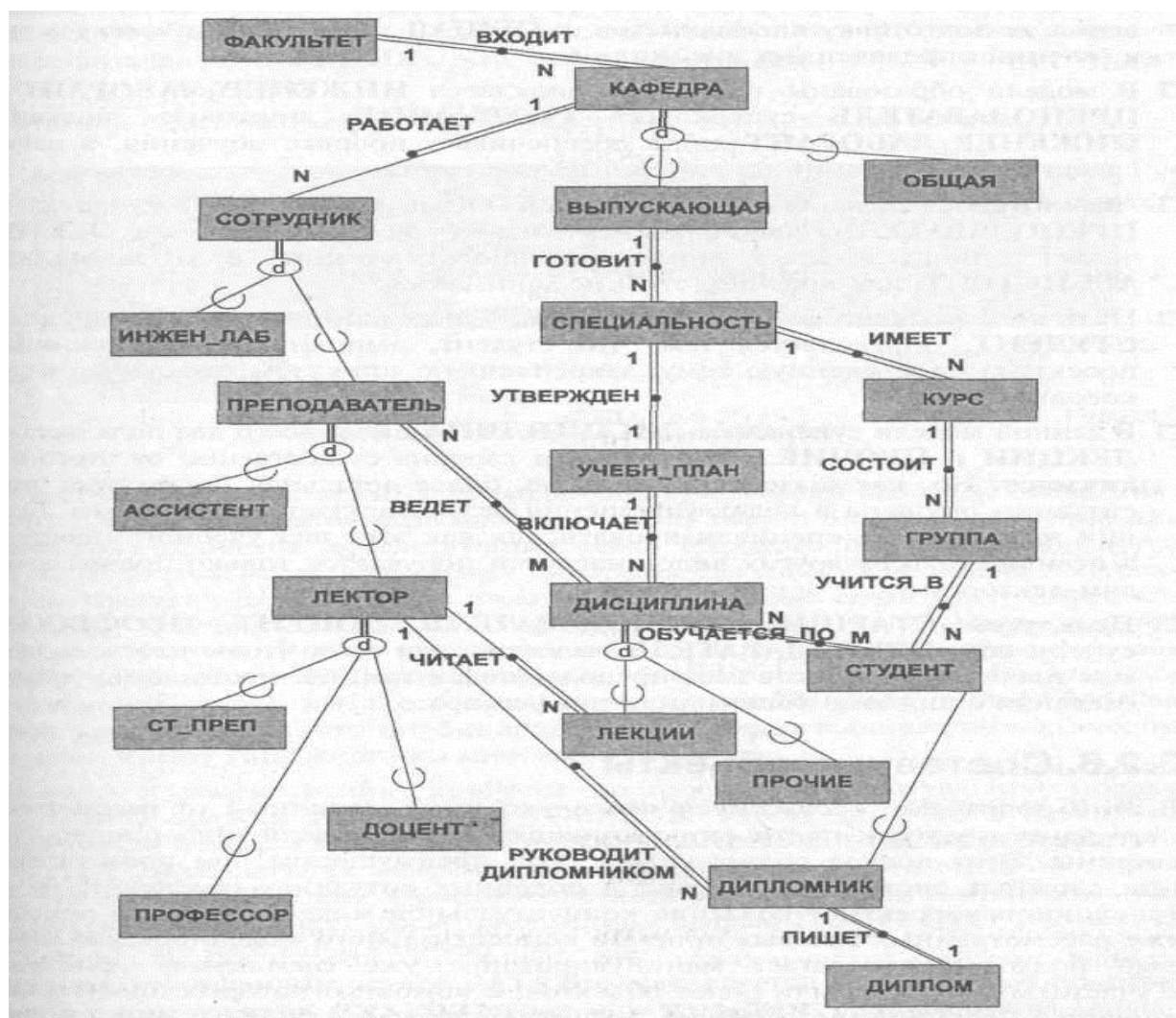


Рисунок 8. ER-діаграма предметної області НАВЧАЛЬНИЙ ПРОЦЕС

У розглянутій моделі проведена специфікація ряду об'єктів з утворенням суперкласів і підкласів.

– Так, наприклад, є суперклас КАФЕДРА і його підкласи ВИПУСКАЮЧА й ЗАГАЛЬНА. Поява зазначених підкласів порозумівається тим, що в них різні функції. ВИПУСКАЮЧА відповідальна за підготовку фахівців, а ЗАГАЛЬНА тільки бере участь у веденні певних дисциплін.

– У моделі утворені також два підкласи ІНЖЕНЕР_ЛАБОРАНТ і ВИКЛАДАЧ суперкласу СПІВРОБІТНИК, оскільки підклас ІНЖЕНЕР_ЛАБОРАНТ лише забезпечує процес навчання, а безпосередньо у веденні дисциплін не бере участь.

- Аналогічно виділені підкласи ЛЕКТОР й АСИСТЕНТ суперкласу ВИКЛАДАЧ, специфіка діяльності яких різна: ЛЕКТОР читає лекції й веде дипломне проектування, у той час як АСИСТЕНТ, як правило, цим не займається.
- Наявність підкласу ДИПЛОМНИК, виділеної із суперкласу СТУДЕНТ, визначається тим, що студент, що виконує дипломний проект на певну тему, закріплену наказом, має потребу в керівнику.
- У даній моделі суперклас ДИСЦИПЛІНА має всього два підкласи - ЛЕКЦІЇ та ІНШІ, хоча реальна картина істотно від цього відрізняється. Більше детальне пророблення цієї ситуації опущені з метою спрощення результуючої діаграми. Лекції ж довелося спеціалізувати, тому що цей вид навчального процесу на відміну від всіх інших видів навантаження поручається тільки викладачам-лекторам.
- Підкласи СТАРШИЙ_ВИКЛАДАЧ, ДОЦЕНТ, ПРОФЕСОР суперкласу ВИКЛАДАЧ виділені для того, щоб уникнути порожніх значень атрибутів всіх викладачів там, де необхідно вказати відомості про диплом доцента або професори.

2 Складені об'єкти

Тепер розглянемо трохи інший підхід до побудови концептуальної моделі інформаційної системи. Цей підхід виявляє свої переваги при проектуванні складних систем і дозволяє в подібних ситуаціях забезпечити більше успішне й коректне створення концептуальної моделі, чим за допомогою базових прийомів концептуального моделювання. Даний підхід припускає маніпулювання вже складовими об'єктами за участю двох, трьох або більше об'єктів, за допомогою яких простіше моделюються більшість проблем, що виникають як у бізнесі, так й в інших предметних областях у дійсності. Обговоримо основні положення такого підходу. Два зв'язаних об'єкти можна використати як об'єкт. Наприклад, об'єкти ВИКЛАДАЧ і КУРС зі зв'язком ЧИТАЄ становлять нову структуру, що сама по собі є об'єктом. Подібні об'єкти можуть мати атрибути (Аудиторія, Час) і брати участь в інших зв'язках. Іноді складеним об'єктам дають об'єктні

імена - іменники (ЛЕКЦІЯ), що дуже зручно, якщо такий новий об'єкт зв'язується з іншими. Складений об'єкт може бути зображений так, як показано на рисунку 9.

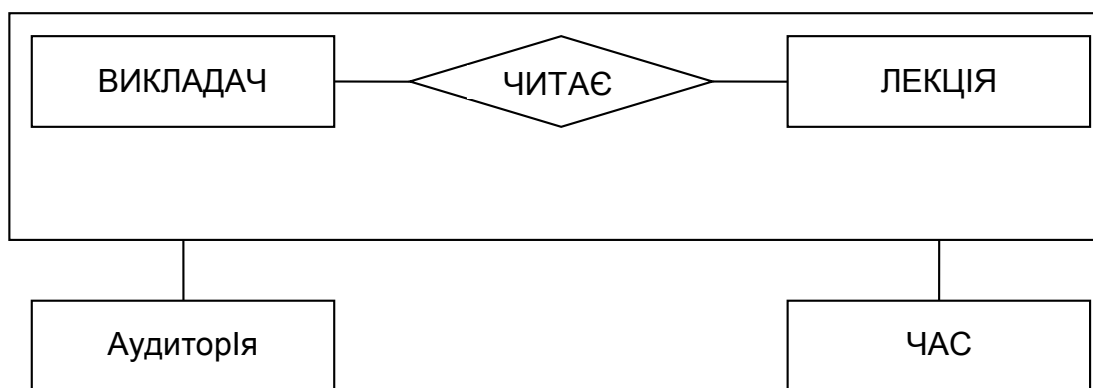


Рисунок 9. Складений об'єкт ЛЕКЦІЯ

В усіх раніше розглянутих прикладах були присутні бінарні зв'язки. Зв'язку трьох і більше об'єктів називаються зв'язками високого порядку. Звичайно для спрощення процесу моделювання зв'язку високого порядку розбивають на послідовність вкладених бінарних зв'язків. Така заміна припустима, але варто врахувати, що при цьому не всі бінарні зв'язки будуть мати який-небудь "фізичний" зміст. Тому іноді при побудові моделей даних для більше точного відбиття процесів у реальному світі доцільніше користуватися N-арними зв'язками. Відповідно до максимальних потужностей зв'язків високого порядку часто використовується перевірене на практиці допущення, що всі бінарні зв'язки, що становлять зв'язок високого порядку, мають тип "багато до багатьох".

Розглянемо процес побудови концептуальної моделі предметної області ПОСТАВКА ТОВАРІВ. Розглянемо його в трохи зміненому виді, а саме ПОСТАВКА ТОВАРІВ НА СКЛАД. Екземпляр об'єкта ТОВАР пов'язаний з екземпляром об'єкта СКЛАД. Цей зв'язок позначений, як НАДІЙШОВ НА. Об'єднання двох зазначених зв'язаних об'єктів в один об'єкт дозволяє зв'язати останній зв'язком НАДІЙШОВ У с об'єктом ДАТА, що дає новий об'єкт (рис. 10).

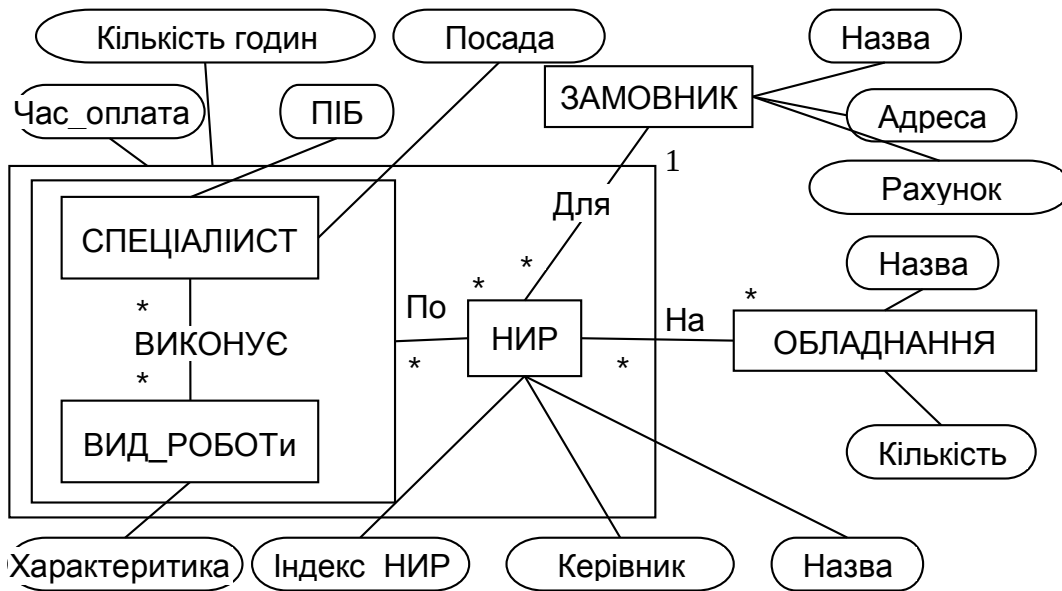


Рисунок 10. Модель предметної області НИР

Об'єкти ФАХІВЕЦЬ і ВИД_РОБОТИ утворюють самий внутрішній складений об'єкт. Вони мають свої атрибути, призначення яких очевидно й не вимагає додаткових роз'яснень, за винятком атрибута Характеристика, що являє собою поле вільного формату, де користувач може записати будь-яку інформацію описового характеру. Зв'язок цих об'єктів має назва ВИКОНУЄ й визначена як "багато до багатьох", тому що передбачається, що той самий фахівець може виконувати різні види робіт, і та сама робота може виконуватися різними фахівцями. Отриманий складений об'єкт у свою чергу має зв'язок з об'єктом НИР. Цей зв'язок визначений також як "багато до багатьох": той самий елемент складеного об'єкта може брати участь у різних НИР, і в одній і тій же НИР можуть брати участь різні екземпляри складеного об'єкта. Об'єкту НИР приписані атрибути: Індекс теми, Назва, Керівник, і він зв'язаний ще із двома об'єктами ЗАМОВНИК й УСТАТКУВАННЯ. Об'єкт ЗАМОВНИК визначає хто, з якою адресою й з яким рахунком оплачує проведені роботи. Природно, та сама тема може мати тільки одного замовника, але ніхто не забороняє деякому замовникові укласти кілька договорів на виконання НИР. Тому зв'язок ДЛЯ кваліфікується як "один до багатьох". Зв'язок НА визначає: по якій НИР яке встаткування й у якій кількості було придбано. Зрозуміло, що здоровий підхід до рішення проблеми вимагає цей зв'язок визначити як "багато до багатьох".

При подальшому аналізі розроблювальної моделі виявляється, що дуже корисно побудований раніше складовий об'єкт об'єднати з об'єктом НИР у новий складений об'єкт. І атрибути Годинна плата, Кількість годин приписати вже новому (зовнішньому) складеному об'єкту, тому що вони визначаються в нашому випадку для кожного фахівця з кожного виду робіт і для кожної НИР, тобто свої в кожного екземпляра зовнішнього складеного об'єкта.

Зображену вище на схемі модель даних, нехай і дуже спрощену, можна використати для одержання безлічі різноманітних звітів. Наприклад, можна створити звіт, що відбиває, які види робіт кожен фахівець виконував для кожного замовника й у який НИР. Можна оцінити цю роботу в годинниках, в отриманій заробітній платі. Можна одержати різноманітні відомості про встаткування й ще багато чого іншого.

Складені об'єкти й зв'язки високого порядку - потужні засоби, що часто застосовуються при моделюванні складних предметних областей.

Питання для самоконтролю знань студентів

- 1 Приведіть приклад концептуальної моделі деякої предметної області.
- 2 Побудуйте концептуальну модель деякої предметної області, використовуючи поняття складеного об'єкта.
- 3 Поясніть поняття спеціалізації, генералізації, категоризації.
- 4 Створіть концептуальну модель даних деякого університету, яка б дозволяла одержати наступну інформацію:
 - кількість викладачів, що працюють на факультеті;
 - список викладачів факультету;
 - дисципліни, досліджувані студентами певної спеціальності в шостому семестрі;
 - обсяг лекцій по дисциплінах другого семестру для певної спеціальності факультету;
 - прізвища викладачів, ведучих вищу математику на факультеті.

План самостійного вивчення теми № 2

з навчальної дисципліни Організація баз даних

Тема: Цілісність даних. Цілісність бази даних

Мета: Ввести поняття цілісності даних. Розглянути класифікацію обмежень цілісності. Пояснити використання декларативні обмежень цілісності
знати:

- види обмежень цілісності даних;
- основні засоби, що забезпечують збереження цілісності даних при створенні та роботі з реляційними базами даних;

Час – 4 години

Навчальні питання:

1 Поняття про обмеження цілісності. Класифікація обмежень.

2 Декларативні обмеження цілісності

2.1 Цілісність відношень

2.2 Цілісність атрибутів

2.3 Цілісність зв'язків між відношеннями

2.4 Цілісність зв'язків між атрибутами

3. Цілісність бази даних

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

По-перше Обмеження цілісності класифікують за: походженням; способом підтримки; терміном перевірки; областю дії; можливістю обмежувати переходи бази даних з одного стану в інший.

По-друге Декларативні обмеження цілісності — це обмеження, що фіксують умови, яким має відповідати база даних. Завдання СКБД - не допускати порушення цих умов. Зазвичай декларативні обмеження цілісності визначаються мовою опису структури даних.

По-третє

За областю дії розрізняють обмеження, що стосуються: відношення; атрибута; зв'язків між відношеннями; зв'язків між атрибутами.

По-четверте

У реляційній СКБД цілісність відношень визначається за допомогою первинного ключа, для якого мають виконуватися такі обмеження цілісності: атрибути первинного ключа не можуть містити NULL-значень; значення первинного ключа (як окремого атрибута або сукупності атрибутів) не можуть дублюватися в межах відношення.

У реляційній СКБД цілісність атрибутів може забезпечуватися: зазначенням типів даних та їх розмірів (обов'язкова властивість); визначенням, чи є обов'язковим значення атрибута (NULL/NOT NULL); визначенням, чи може значення атрибута дублюватися (UNIQUE); зміною значення атрибута після його введення; встановленням умов, яким мають відповідати значення атрибута.

Цілісність зв'язків між атрибутами визначається умовою, якій має відповідати сукупність атрибутів одного або кількох відношень. Такі умови специфікуються тригерами.

Література

- 1 Гайна Г. А. Основи проектування баз даних: навч. посіб. . [Текст] /Київ: КНУБА, 2005. 204 с.
- 2 Трофименко О. Г. Організація баз даних :Навч. посібник . [Текст] / О. Г. Трофименко, Ю. В. Прокоп, Н. І. Логінова, І. М. Копитчук. 2-ге вид. виправ. і доповн— Одеса : Фенікс, 2019. – 246 с.
- 3 Харів Н. О. Базы даних та інформаційні системи: навчальний посібник . [Текст] / Н. О. Харів. – Рівне : НУВГП, 2018. – 127 с.
- 4 Ярцев В.П. Організація баз даних та знань: навчальний посібник. [Текст] / В.П. Ярцев – К. ДУТ 2018.-214с.

Навчальні матеріали до самостійного вивчення теми

Терміном *цілісність* даних позначають достовірність і точність інформації, що зберігається в базі. Цілісність досягається забезпеченням відповідності даних певним додатковим обмеженням, крім тих, які накладаються схемою бази на структуру даних та їхні типи.

Обмеження цілісності – це правила, які обмежують усі можливі стани бази даних, а також переходи з одного стану в інший. Таким чином, обмеження цілісності визначають множину «допустимих» станів і переходів між ними. База даних перебуває в цілісному стані, якщо вона відповідає всім визначеним для неї вимогам цілісності.

1 Поняття про обмеження цілісності. Класифікація обмежень.

Обмеження цілісності класифікують за:

- походженням;
- способом підтримки;
- терміном перевірки;
- областю дії;
- можливістю обмежувати переходи бази даних з одного стану в інший.

За походженням обмеження цілісності поділяють на:

- структурні;
- семантичні.

Структурні обмеження цілісності – це обмеження, які впливають із властивостей структури даних, що зберігаються в базі.

Семантичні обмеження цілісності – це обмеження, що накладаються предметною областю, яка моделюється.

За способом підтримки виділяють такі класи обмежень цілісності:

- декларативні;
- процедурні.

Декларативні обмеження цілісності — це обмеження, що фіксують умови, яким має відповідати база даних. Завдання СКБД - не допускати порушення цих умов. Зазвичай декларативні обмеження цілісності визначаються мовою опису структури даних. Прикладом такого обмеження є: «Фонд зарплати факультету має бути рівним сумі фондів зарплати всіх його кафедр».

Декларативні обмеження цілісності специфікуються фразою
CONSTRAINT в описі таблиці або її полів.

Процедурні обмеження цілісності — це описи дій, спрямованих на забезпечення цілісності. Прикладом такого обмеження є: «Під час зміни фонду зарплати будь-якої з кафедр автоматично змінити фонд зарплати факультету так, щоб він дорівнював сумі фондів зарплати усіх кафедр».

Процедурні обмеження цілісності специфікуються тригерами.

За часом перевірки обмеження цілісності поділяються на такі, що:

- перевіряються негайно;
- мають відкладену перевірку.

Обмеження, що перевіряються негайно, перевіряються безпосередньо у момент виконання операції, яка може порушити цілісність. Наприклад, неповторюваність назви факультету перевіряється під час додавання нового рядка до таблиці, яка містить інформацію про факультети, або під час заміни імені факультету в існуючому рядку таблиці. Якщо обмеження порушується, то операція блокується.

Обмеження цілісності з відкладеною перевіркою використовуються у тому випадку, коли для підтримання бази даних у несуперечному стані потрібно виконати дві або більше операції. Прикладом обмеження цілісності, яке не може бути перевірено негайно, є декларативне обмеження щодо фондів заробітної плати факультету та його кафедр. Після додавання нової кафедри з заданим фондом фінансування база даних переходить у суперечний стан, оскільки фонд фінансування факультету стає не рівним сумі фондів фінансування його кафедр. Для того щоб повернути базу даних у несуперечний стан, потрібно виконати ще одну операцію — оновити фонд фінансування факультету. У такому випадку ці дві операції об'єднуються в одну транзакцію і обмеження цілісності перевіряється після її завершення.

За областю дії розрізняють обмеження, що стосуються до:

- відношення;
- атрибута;
- зв'язків між відношеннями;
- зв'язків між атрибутами.

За можливістю обмежувати переходи бази даних з одного стану в інший обмеження цілісності поділяються на:

- статичні;
- динамічні.

Статичні обмеження цілісності накладають обмеження на можливі стани бази даних. Наприклад, декларативні обмеження цілісності, що описуються далі, є статичними.

Динамічні обмеження цілісності задають обмеження на можливі переходи бази даних з одного стану в інший.

2 Декларативні обмеження цілісності

Під час розгляду декларативних обмежень цілісності постають такі питання:

- на які об'єкти моделі даних поширюються обмеження цілісності;
- якими є ці обмеження;
- як ті чи інші обмеження цілісності специфікуються;
- які існують механізми підтримання цілісності.

У реляційних базах даних до об'єктів, на які поширюються обмеження цілісності, належать такі:

- відношення;
- атрибути;
- зв'язки між відношеннями;
- зв'язки між атрибутами.

У сучасних СКБД є й багато інших об'єктів бази даних, щодо яких специфікуються обмеження цілісності.

Розглянемо, як задаються обмеження цілісності для об'єктів реляційних СКБД. Специфікація буде подана мовою PL/SQL, що застосовується в СКБД Oracle і в якій для

специфікації структурних обмежень цілісності використовується фраза CONSTRAINT (складова частина речень CREATE TABLE і ALTER TABLE).

2.1 Цілісність відношень

У реляційній СКБД цілісність відношень визначається за допомогою первинного ключа, для якого мають виконуватися такі обмеження цілісності:

- атрибути первинного ключа не можуть містити NULL-значень;
- значення первинного ключа (як окремого атрибута або сукупності атрибутів) не можуть дублюватися в межах відношення.

Цілісність за первинним ключем специфікується так:

CONSTRAINT <ім'я обмеження> PRIMARY KEY (<перелік полів>)

де <перелік полів> — поля, що складають первинний ключ.

Ця специфікація вказує, які саме поля є ключовими.

Для специфікації можливих ключів використовується фраза UNIQUE.

Механізм підтримки цілісності відношень реалізує СКБД

```
CREATE TABLE КАФЕДРА
```

```
( #D NUMBER(2),
```

```
Назва VARCHAR2(9),
```

```
Завідувач VARCHAR2(10),
```

```
CONSTRAINT DepPK PRIMARY KEY (#D) );
```

2.2 Цілісність атрибутів

У реляційній СКБД цілісність атрибутів може забезпечуватися:

- зазначенням типів даних та їх розмірів (обов'язкова властивість);
- визначенням, чи є обов'язковим значення атрибута (NULL/NOT NULL);
- визначенням, чи може значення атрибута дублюватися (UNIQUE);
- зміною значення атрибута після його введення;
- встановленням умов, яким мають відповідати значення атрибута.

Для похідних атрибутів цілісність має гарантуватися процедурою їхнього обчислення. Цілісність атрибутів специфікується фразами NULL/NOT NULL і UNIQUE у визначенні атрибута (поля).

Якщо унікальними мають бути значення не окремих полів, а сукупності полів, то це вказується так:

```
CONSTRAINT <ім'я обмеження> UNIQUE (<перелік полів>)
```

Цілісність атрибутів специфікується також зазначенням обмеження

```
CONSTRAINT <ім'я обмеження> CHECK (<умова>)
```

Умова визначається на атрибутах відношення. Фраза CHECK може також вказуватися в означенні атрибута.

```
CREATE TABLE ГРУПА ( #GNUMBER(3),
```

```
#D NUMBER(3).
```

```
Курс NUMBER(1) CHECK (Course IN(1.2.3.4.5)).
```

```
Номер NUMBER(3) CHECK (Номер > 0).
```

```
Кількість NUMBER(2) CHECK (Кількість > 0).
```

```
#Куратор NUMBER(3) UNIQUE.
```

```
CONSTRAINT GrpPK PRIMARY KEY (#G).
```

```
CONSTRAINT GrpUNQ UNIQUE (Курс. Номер))
```

Зазначимо, що коли UNIQUE вказується для групи атрибутів, то йдеться про унікальність саме групи атрибутів, кожний з них окремо не обов'язково має бути унікальним. За допомогою фрази UNIQUE специфікуються також можливі ключі таблиці, тоді вона має використовуватися разом з обмеженням NOT NULL. Механізм підтримки цілісності атрибутів реалізує СКБД.

2.3 Цілісність зв'язків між відношеннями

Цілісність зв'язків між відношеннями визначається зовнішніми ключами. Під час специфікації зовнішнього ключа вказується відповідний йому первинний ключ іншого відношення. Такий первинний ключ називається референційним.

Відношення, на яке здійснюється посилання за допомогою зовнішнього ключа, називається батьківським, а те, яке посилається, — дочірнім.

Якщо зовнішній ключ певного відношення може містити NULL значення, це означає, що зв'язок даного відношення з іншим є факультативним. NOT NULL- специфікація стовпців зовнішнього ключа свідчить, що зв'язок є обов'язковим. Означення зовнішнього ключа свідчить про наявність цілісності за посиланням: значення зовнішнього ключа має посылатися на значення первинного ключа іншого відношення. Тобто ситуація, коли зовнішній ключ має значення, відсутнє серед значень відповідного первинного ключа, розглядається як ситуація, що порушує цілісність за посиланням.

Цілісність за посиланням в Oracle специфікується так:

```
CONSTRAINT <назва обмеження> FOREIGN KEY (<перелік полів 1>)  
REFERENCES <ім'я таблиці>(<перелік полів 2>)  
[ON DELETE {CASCADE | SET NULL}]
```

де «перелік полів 1» — поля, що становлять зовнішній ключ, «ім'я таблиці» — ім'я таблиці референційного ключа, «перелік полів 2» — поля, з яких складається референційний ключ. В Oracle допускається, щоб «перелік полів 2» був не первинним ключем, а списком довільних полів, які мають специфікацію UNIQUE.

Механізм підтримки цілісності за посиланням реалізує СКБД.

Розглянемо, як діє механізм під час видалення рядка з батьківської таблиці.

Можливі три варіанти:

- рядок батьківської таблиці може бути видалений лише в тому випадку, якщо немає зовнішніх ключів, що посилаються на значення референційного ключа цього рядка;
- під час видалення рядка батьківської таблиці видаляються також усі рядки в інших таблицях, значення зовнішніх ключів яких посилаються на значення референційного ключа цього рядка (каскадне видалення);
- під час видалення рядка батьківської таблиці всі посилання на значення видаленого референційного ключа замінюються на NULL.

Відсутність фрази [ON DELETE {CASCADE | SET NULL}] вказує на перший варіант. Фраза ON DELETE CASCADE в специфікації референційної цілісності вказує на другий варіант, а ON DELETE SET NULL - на третій.

2.4 Цілісність зв'язків між атрибутами

Цілісність зв'язків між атрибутами визначається умовою, якій має відповідати сукупність атрибутів одного або кількох відношень. Такі умови специфікуються тригерами.

Наприклад, аби вказати, що кількість студентів на лекції не може перевищувати кількість місць в аудиторії, слід означити такий тригер:

```
CREATE TRIGGER ЛЕКЦІЯ ВСТАВКА ОНОВЛЕННЯ
BEFORE INSERT, UPDATE ON ЛЕКЦІЯ WHEN
(SELECT Місця
FROM АУДИТОРІЯ
WHERE АУДИТОРІЯ.#R = ЛЕКЦІЯ.#R) <
(SELECT Кількість
FROM ГРУПА
WHERE ГРУПА.#С = ЛЕКЦІЯ.#С)
BEGIN
ROLLBACK TRANSACTION
END
```

Обмеження цілісності запобігають появі в БД суперечливих даних [1]. Вирішення цієї проблеми на стадії проектування полягає в такому:

- наявність обов'язкових і необов'язкових значень даних для атрибутів;
- наявність обмежень для доменів атрибутів (визначення області значень або діапазону значень);
- цілісність сутностей (обов'язкова наявність Primary Key в кожному відношенні);
- посилкова;
- обмеження предметної галузі (бізнес-правила), що реалізуються як засобами БД, так і на рівні застосувань.

3. Цілісність бази даних

На зв'язки між відношеннями накладаються певні правила цілісності, які допомагають зберігати базу даних в стабільному і робочому стані. Помилкова ситуація може виникати при додаванні, зміні чи видаленні кортежів з відношень. Якщо дія, що виконується, суперечить правилам цілісності бази даних, то СКБД видає повідомлення про помилку і виконання дії відхиляється, в іншому випадку – отримується позитивний результат.

1 Додавання кортежів. Відбувається спочатку у головне відношення. Додавання кортежів у підпорядковане відношення з новим зовнішнім ключем приводить до помилки.

2 Модифікація кортежів. Найчастіше пов'язана із зміною первинного і зовнішнього ключів. При цьому можливі три операції:

а. Без будь-якої дії (NO ACTION). Модифікація первинного ключа відбувається у головному відношенні, якщо у підпорядкованому відношенні немає відповідних кортежів. При наявності таких їх потрібно видалити, бо інакше це приведе до помилкової ситуації.

б. Каскадування (CASCADE). Модифікація первинного ключа у головному відношенні призводить до відповідної зміни зовнішніх ключів у підпорядкованому відношенні.

в. NULL-значення (SET NULL). Модифікація первинного ключа у головному відношенні призводить до встановлення NULL-значення у зовнішніх ключах підпорядкованого відношення.

3 Видалення кортежів. При цьому можливі три операції:

а. Без будь-якої дії (NO ACTION). Видалення кортежів з головної таблиці неможливе, якщо є відповідні кортежі у підпорядкованому відношенні.

б. Каскадування (CASCADE). Видалення кортежів у головному відношенні призводить до видалення відповідних кортежів у підпорядкованому відношенні.

в. NULL-значення (SET NULL). Видалення кортежів у головному відношенні призводить до встановлення NULL-значення у зовнішніх ключах підпорядкованого відношення.

Під цілісністю даних (посилальною цілісністю даних) у загальному значенні розуміють вимоги несуперечності представленої в БД інформації, щоб в БД не було посилань на неіснуючі дані, і не було зв'язаних даних, посилання на які відсутні (загублені дані).

Основним засобом забезпечення цілісності даних є створення зв'язків між таблицями БД. Зв'язки між таблицями відповідають зв'язкам між відповідними сутностями і відношеннями, дані про які представлені в цих таблицях. Так само, як і зв'язки між відношеннями, зв'язки між таблицями можуть бути зв'язками типу “1:1”, “1:М” і “М:М”. Зв'язок між таблицями встановлюється через зовнішні ключі (ключі зв'язку). У зв'язаних таблицях у зв'язаних записах значення полів, що є ключами зв'язку, повинні співпадати.

Розглянемо приклад зв'язаних таблиць. На рис. 1 показано дві таблиці: ТОВАРИ і ПОКУПКИ, між якими встановлений зв'язок типу “1:М”.

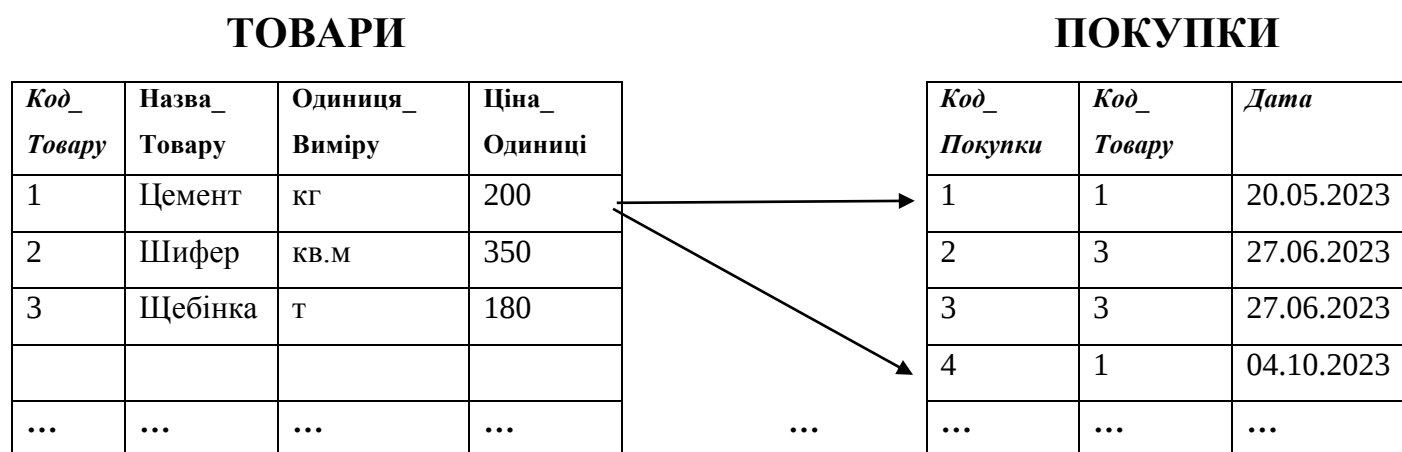


Рисунок 1. Зв'язок типу “1:М” між таблицями ТОВАРИ і ПОКУПКИ

Зв'язок створено по полю (ключу зв'язку) КодТов. У таблиці ТОВАРИ поле Код_Товару є первинним ключем і одночасно ключем зв'язку. У таблиці ПОКУПКИ поле Код_Товару є ключем зв'язку, а первинним ключем є поле Код_Покупки. У даному випадку прийнято таблицю ТОВАРИ називати головною таблицею, а таблицю ПОКУПКИ – підлеглою.

Цілісність даних забезпечується виконанням наступних вимог:

- у підлеглу таблицю не можна додати запис з неіснуючим у головній таблиці значенням ключазв'язку;
- у головній таблиці не можна видалити запис, якщо в підлеглій таблиці є зв'язані з нею записи;

- не можна змінити значення ключа зв'язку, якщо в зв'язаній таблиці є зв'язані записи.

Зв'язаними називають такі записи, у яких ключі зв'язку мають однакові значення.

У розглянутому прикладі цілісність даних буде порушена, якщо з таблиці ТОВАРИ видалити запис товару “Цемент”, чи додати в таблицю ПОКУПКИ запис для якого-небудь товару, для якого ще не створений запис у таблиці ТОВАРИ. Цілісність даних також була б порушена, якщо в таблиці ПОКУПКИ змінити значення в поле Код_Товару.

Якщо користувач спробує виконати дії, що порушують цілісність даних, СУБД не дозволить це зробити і видасть повідомлення про неприпустимість даної дії. В усіх СУБД маються спеціальні засоби для автоматизації деяких операцій по підтримці цілісності даних. Так, передбачена можливість при створюванні зв'язків між таблицями задати вимогу автоматичного виконання таких операцій:

- при зміні значень у полях зв'язку головної таблиці автоматично змінювати значення зв'язаних полів у підлеглих таблицях (каскадне відновлення зв'язаних полів);
- при видаленні запису в головній таблиці автоматично видаляти зв'язані записи в підлеглих таблицях (каскадне видалення зв'язаних записів).

Питання для самоконтролю знань студентів

- 1 Що таке обмеження цілісності в базі даних?
- 2 Як використовують обмеження цілісності БД?
- 3 Яким чином відбувається налаштування обмежень цілісності в базі даних?
- 4 Для чого використовують обмеження цілісності в БД?
- 5 До яких типів обмежень цілісності відносять:
 - обмеження унікальності;
 - зовнішні ключі;
 - перевірочні обмеження;
 - обмеження на видалення та оновлення;
 - внутрішні ключі;
 - обмеження доступу до даних для певних користувачів.
- 6 У чому полягає перевірка підтримки цілісності?

План самостійного вивчення теми № 3

з навчальної дисципліни Організація баз даних

Тема Тригери: особливості використання

Мета Пояснити особливості використання тригерів. Навчити організовувати роботу з тригерами

знати:

- поняття тригера та класифікацію тригерів;
- поняття тригерного предикату та псевдозапису;

вміти: виконувати:

- створення тригеру;
- активізацію тригеру;
- виключення тригеру;
- видалення тригера.

Час – 5 годин

Навчальні питання:

- 1 Тригери бази даних. Призначення тригерів.
 - 1.1 Створення та включення тригерів.
 - 1.2 Класифікація тригерів. Порядок активізації тригерів
 - 1.3 Тригерні предикати
 - 1.4 Псевдозаписи.
 - 1.5 Включення, виключення і видалення тригерів.
- 2 Приклади створення тригерів.

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

По-перше Тригер (trigger) — це окрема збережена процедура, яка виконується після настання певних подій у реляційній базі даних без участі користувача. До подій відносяться операції додавання INSERT, видалення рядка в таблиці DELETE та зміна даних у певному стовпці UPDATE.

По-друге

Тип тригера визначається поєднанням наступних трьох параметрів: характером впливу тригера на рядки пов'язаної з ним таблиці (строко-вий або операторний); моментом запуску тригера: до (BEFORE) або після (AFTER) виконання активізуючого тригер оператора; типом активізуючого тригер оператора (INSERT, DELETE, UPDATE).

По-третє

Для тригерів, що працюють з кортежами (рядками) таблиць бази даних, існують спеціальні конструкції, які дозволяють при виконанні операторів над рядком таблиці звертатися як до старих значень, які знаходились в ній до модифікації, так і до нових, які з'являються в рядку після її модифікації. Ці конструкції називаються псевдозаписами і позначаються old і new.

По-четверте

Тригери можна використовувати:

- для реалізації складних обмежень цілісності даних, які неможливо реалізувати через декларативні обмеження, що встановлюються при створенні таблиці;
- для контролю за інформацією, що зберігається в таблиці, за допомогою реєстрації внесених змін і користувачів, які виконують ці зміни;
- для автоматичного оповіщення інших програм про те, що необхідно робити в разі зміни інформації, що міститься в таблиці;
- для публікації інформації про різні події в середовищі "публікація– підписка".

Література

- 1 Трофименко О. Г. Організація баз даних : Навч. посібник . [Текст] / О. Г. Трофименко, Ю. В. Прокоп, Н. І. Логінова, І. М. Копитчук. 2-ге вид. виправ. і доповн. – Одеса : Фенікс, 2019. – 246 с.
- 2 Харів Н. О. Базы даних та інформаційні системи: навчальний посібник . [Текст] / Н. О. Харів. – Рівне : НУВГП, 2018. – 127 с.
- 3 Ярцев В.П. Організація баз даних та знань: навчальний посібник. [Текст] / В.П. Ярцев – К. ДУТ 2018.-214с.

Навчальні матеріали до самостійного вивчення теми

1. Тригери бази даних. Призначення тригерів.

Тригер бази даних – це оформлений спеціальним чином іменований блок що зберігається в БД. Кожен тригер пов'язаний з визначеною таблицею і автоматично запускається при виконанні одного з (INSERT, DELETE, UPDATE) або їх сукупності над цією таблицею.

Тригери можуть бути використані:

- для реалізації складних обмежень цілісності даних, які не можуть бути здійснені стандартним чином при створенні таблиці;
- запобігання невірних транзакцій;
- виконання процедур комплексної перевірки прав доступу та секретності даних;
- генерації деяких виразів на основі значень, наявних у стовпцях таблиць;
- при реалізації складних бізнес-правил для обробки даних (можливість відстежити оновлення даних у зв'язаних таблицях при зміні в одній з них).

1.1 Створення і включення тригерів

Для створення і автоматичного включення тригера застосовується такий загальний синтаксис:

```
CREATE [OR REPLACE] TRIGGER ім'я_тригера  
{BEFORE | AFTER}  
{INSERT | DELETE | UPDATE [OF список_стовпців]}  
ON ім'я_таблиці [FOR EACH ROW] [WHEN умова]
```

При наявності ключових слів OR REPLACE тригер створюється заново, якщо він вже існує.

Конструкція BEFORE | AFTER вказує на момент запуску тригера. Варіант BEFORE означає, що тригер буде запускатися перед виконання активізуючого оператора; варіант AFTER означає, що тригер буде запускатися після виконання активізуючого оператора.

Конструкція INSERT | DELETE | UPDATE [OF список_стовпців] вказує тип активізуючого тригер DML-оператора. Дозволяється, використовуючи логічну операцію OR, задати сукупність активізуючих операторів, наприклад INSERT OR DELETE. Якщо при використанні варіанта UPDATE вказаний список стовпців, то тригер буде запускатися при модифікації одного із зазначених стовпців; якщо список стовпців відсутній, то тригер буде запускатися при зміні будь-якого із стовпців пов'язаної з тригером таблиці.

Конструкція FOR EACH ROW вказує на характер впливу тригера: строковий або операторний. Якщо конструкція FOR EACH ROW присутня, то тригер є строковим; при відсутності її тригер є операторним. Операторний тригер запускається один раз до або після виконання активізуючого тригер DML-оператора незалежно від того, скільки рядків у пов'язаної з тригером таблиці піддалось модифікації. Строковий тригер запускається один раз для кожного з рядків, які піддаються модифікації DML-оператором, активізуючого тригер.

За допомогою ключового слова WHEN можна задати додаткове обмеження на рядки пов'язаної з тригером таблиці, при модифікації яких може бути запущений тригер.

1.2 Класифікація тригерів. Порядок активізації тригерів

В основному розрізняють дванадцять типів тригерів. Тип тригера визначається поєднанням наступних трьох параметрів:

- характером впливу тригера на рядки пов'язаної з ним таблиці (строковий або операторний);
- моментом запуску тригера: до (BEFORE) або після (AFTER) виконання активізуючого тригер DML-оператора;
- типом активізуючого тригер DML-оператора (INSERT, DELETE, UPDATE).

Якщо у таблиці є декілька типів тригерів, то вони активізуються за такою схемою:

- виконується операторний тригер BEFORE (якщо їх декілька, то нічого про порядок їх виконання сказати не можна);
- виконується строковий тригер BEFORE;
- виконується активізуючий тригер DML-оператор з наступною перевіркою всіх обмежень цілісності даних;
- виконується строковий тригер AFTER з наступною перевіркою всіх обмежень цілісності даних;
- виконується операторний тригер AFTER.

1.3 Тригерні предикати.

Якщо в тригері вказується сукупність активізуючих тригер операторів (наприклад, INSERT OR DELETE), то для розпізнавання того, який конкретно з операторів виконується над пов'язаною з тригером таблицею, використовуються тригерні предикати: INSERTING, DELETING, UPDATING. Вони представляють собою логічні функції, які повертають TRUE, якщо тип активізуючого оператора збігається з типом предиката, і FALSE – в протилежному випадку. Для завдання одних і тих же дій у разі виконання різних DML-операторів в умовному операторі тригерні предикати об'єднуються за допомогою логічних операцій.

1.4 Псевдозаписи.

Для тригерів, що працюють з кортежами (рядками) таблиць бази даних, існують спеціальні конструкції, які дозволяють при виконанні ператорів над рядком таблиці звертатися як до старих значенням, які знаходились в ній до модифікації, так і до нових, які з'являться в рядку після її модифікації. Ці конструкції називаються псевдозаписами і позначаються old і new. Структура цих псевдозаписів ідентична структурі рядка модифікуємої таблиці, але оперувати можна тільки окремими полями псевдозапису. Звернення до полів псевдозапису відбувається за такою схемою: перед old або new ставиться символ двокрапка (:), далі через точку вказується назва поля. Значення,

які беруть поля псевдозапису при виконанні активізуючих операторів, визначаються таким чином.

Оператор INSERT – псевдозапис: new еквівалентний рядку, що вставляється, а псевдозапис: old у всіх полях має значення NULL.

Оператор DELETE – псевдозапис: old еквівалентний рядку, що видаляється, а псевдозапис: new у всіх полях має значення NULL.

Оператор UPDATE – псевдозапис: new еквівалентний рядку, отриманому в результаті модифікації, а псевдозапис: old у всіх полях має початкове значення рядка.

1.5 Включення, виключення і видалення тригерів.

Зберігаємий в БД тригер можна тимчасово відключити, не видаляючи його з БД. Для цього використовується наступна команда:

```
ALTER TRIGGER ім'я_тригера DISABLE;
```

Включити тригер через деякий проміжок часу можна, використовуючи команду

```
ALTER TRIGGER ім'я_тригера ENABLE;
```

Заборонити або дозволити запуск всіх тригерів, пов'язаних з деякою таблицею, можна за допомогою команди

```
ALTER TABLE ім'я_таблиці  
{DISABLE | ENABLE} ALL TRIGGERS;
```

де варіант DISABLE використовується для відключення, а варіант ENABLE – для включення всіх тригерів даної таблиці.

Видалення тригера з БД здійснюється за допомогою команди:

```
DROP TRIGGER ім'я_тригера;
```

Інформацію про тригери, які зберігаються в БД, можна отримати з подання словника даних USER_TRIGGERS, наприклад, такою командою:

```
SELECT * FROM USER_TRIGGERS;
```

2 Приклади створення тригерів

Створити тригер, який перед вставкою чергового рядка в таблицю BOOKS_DELIVERY перевіряє наявність зазначеного коду книги в таблиці BOOKS. При відсутності зазначеного коду книги в таблиці BOOKS має генеруватися виняток з видачею відповідного повідомлення.

Додавання нових рядків таблицю BOOKS_DELIVERY виконується оператором INSERT. Оскільки тригер повинен запускатися перед виконанням кожного оператора INSERT, отже, він повинен бути строковим BEFORE-тригером. Для збереження цілісності даних необхідно перевірити, чи є коди книг в таблиці BOOKS, які мають бути внесені. Для цього за допомогою однорядкового оператора SELECT виконується вибірка інформації з таблиці BOOKS, де в умові вибірки використовується поле CODE_BOOK псевдозапису: new. Якщо кількість рядків з даними кодом книги в таблиці BOOKS виявиться рівним нулю, буде згенеровано виняток і видано відповідне повідомлення.

Створення тригера TR1 виконується введенням наступного оператора:

```
CREATE OR REPLACE TRIGGER TR1
BEFORE INSERT ON BOOKS_DELIVERY
FOR EACH ROW
DECLARE
QUANTITY NUMBER (4);
BEGIN
SELECT COUNT (*) INTO QUANTITY FROM BOOKS
WHERE CODE_BOOK =: NEW.CODE_BOOK;
IF QUANTITY = 0 THEN RAISE_APPLICATION_ERROR
(-20212, 'У таблиці BOOKS немає інформації про дану книгу');
END IF;
END TR1;
```

При створенні тригера за допомогою SQL*PLUS, необхідно вказувати в рядку, наступному за останнім оператором, косу риску (/), щоб оператор CREATE ... TRIGGER виконався.

Дія тригера TR1 може бути перевірено виконанням наступного оператора, що здійснює вставку рядка в таблицю BOOKS_DELIVERY і тим самим викликає активізацію тригера TR2:

```
INSERT INTO BOOKS_DELIVERY VALUES  
(21, 15, 'Іванов І. І.', 15, '20 -01-06');
```

Оскільки код книги 15 відсутній в таблиці BOOKS, то буде згенеровано виняток і видано відповідне повідомлення.

2. Створити тригер, який забороняє вносити в таблицю BOOKS рядки зі значенням поля PRICE більше, ніж 500 грн., а також збільшувати ціни книг, інформація про яких зберігається в таблиці BOOKS, більш ніж на 20%. При порушенні даної вимоги має генеруватися виняток з видачею відповідного повідомлення.

Так як внесення нових рядків таблицю BOOKS здійснюється в результаті виконання оператора INSERT, а значення поля PRICE в таблиці BOOKS, що містить ціну книги, може бути змінено в результаті виконання оператора UPDATE, то в тригері вказується сукупність активізуючих операторів. Оскільки тригер повинен запускатися перед виконанням кожного із зазначених операторів, він є рядковим BEFORE-тригером. Так як дії, виконувані тригером, різні для кожного з активізуючих операторів, що модифікують таблицю BOOKS, то для розпізнавання типу оператора використовуються відповідні тригерні предикати INSERTING і UPDATING. Внаслідок того що при вставці нових рядків перевірка має бути піддано нове значення поля PRICE, а при модифікації значення поля PRICE нове значення повинно порівнюватися зі старим значенням, необхідно використовувати псевдозаписи: new і old.

Створення тригера TR2 виконується введенням наступного оператора:

```
CREATE OR REPLACE TRIGGER TR2  
BEFORE INSERT OR UPDATE OF PRICE ON BOOKS  
FOR EACH ROW  
BEGIN  
IF INSERTING THEN  
IF: NEW.PRICE > 5000 THEN  
RAISE_APPLICATION_ERROR
```

```

(-20102, 'У таблицю BOOKS не можна вносити записи з ціною книги > 500');
END IF;
END IF;
IF UPDATING THEN
IF: NEW.PRICE>: OLD.PRICE * 1.2 THEN
RAISE_APPLICATION_ERROR
(-20103, 'У таблиці BOOKS не можна змінювати ціну книги більш ніж на 20%
');
END IF;
END IF;
END TR2;

```

Дія тригера TR2 може бути перевірена виконанням таких операторів, які, здійснюючи вставку рядків таблицю BOOKS і оновлення рядків у таблиці BOOKS, тим самим викликають його активізацію.

Оператор вставки рядків в таблицю BOOKS, що активізує тригер TR2:

```

INSERT INTO BOOKS VALUES
(21, 'Дюна', 'Герберт Ф.', 5268, 'Аст', 'Фантастика');

```

Оператор відновлення рядків в таблиці BOOKS, що активізує тригер TR2:

```

UPDATE BOOKS SET PRICE = 600;

```

Оскільки ці оператори порушують вимоги, пропоновані до значенням і модифікації ціни книг, то у всіх випадках буде згенеровано виняток і видано відповідне повідомлення.

Висновки

Тригери схожі на процедури і функції тим, що також є іменованими блоками PL / SQL і мають розділ оголошень, виконуваний розділ і розділ обробки виняткових ситуацій. Подібно модулям, тригери зберігаються як автономні об'єкти в базі даних і не можуть зберігатися локально в блоці або модулі. Процедура викликається явним чином з іншого блоку, при виклику їй можуть передаватися різні аргументи. Тригер ж

виконується неявно всякий раз, коли відбувається активізуюча його подія, і тригер не має аргументів. Процес виконання тригера називається його активізацією (firing). Подією, що запускає тригер, є операція DML (INSERT, UPDATE або DELETE), виконувана над таблицею або поданням бази даних. У OracleSi ці функції розширені: тригер може спрацьовувати на системну подію, наприклад на запуск або зупинку бази даних, а також на певні види операцій DDL.

Тригери можна використовувати:

- для реалізації складних обмежень цілісності даних, які неможливо реалізувати через декларативні обмеження, що встановлюються при створенні таблиці;
- для контролю за інформацією, що зберігається в таблиці, за допомогою реєстрації внесених змін і користувачів, які виконують ці зміни;
- для автоматичного оповіщення інших програм про те, що необхідно робити в разі зміни інформації, що міститься в таблиці;
- для публікації інформації про різні події в середовищі "публікація– підписка".

Питання та завдання для самоконтролю знань студентів

- 1 Що таке тригер?
- 2 Які є види тригерів?
- 3 Чи може тригер використовуватись як окрема програма?
- 4 До яких дій по відношенню до таблиць приводить виконання тригерів?
- 5 Які оператори використовуються в тригерах?
- 6 Що означає слово POSITION 0 в описовій частині тригера?
- 7 Чи буде виконуватись тригер в описовій частині якого вказане слово
- 8 INACTIVE?
- 9 Який зміст мають змінні NEW та OLD в тілі тригера?

План самостійного вивчення теми № 4

з навчальної дисципліни Організація баз даних

Тема Розподілені бази даних

Мета Ввести поняття розподілених баз даних, розглянути їх логічну архітектуру та архітектуру програмно-технічних засобів розподілених СКБД

знати:

- відмінності між розподіленими системами баз даних, засобами розподіленої обробки даних та паралельними системами баз даних;
- класифікацію розподілених баз даних;
- переваги та недоліки систем керування розподіленими базами даних;
- принципи створення розподілених БД

;Час – 5 годин

Навчальні питання:

1 Концепція розподілених баз даних.

1.1 Основні поняття

1.2 Відмінності між розподіленими системами баз даних, засобами розподіленої обробки даних та паралельними системами баз даних

1.3 Класифікація розподілених баз даних

1.4 Переваги та недоліки систем керування розподіленими базами даних

1.5 Принципи створення розподілених БД

Методичні рекомендації

Самостійно вивчаючи запропоновану тему, студент повинен засвоїти такі основні положення:

По-перше

Розподілена база даних – це сукупність логічно зв'язаних баз даних або частин однієї бази, які розпаралелені між декількома територіально-розподіленими ПЕОМ і забезпечені відповідними можливостями для управління цими базами або їх частинами. Тобто, розподілена база даних реалізується на різних просторово розосереджених обчислювальних засобах, разом з організаційними, технічними і програмними засобами її створення і ведення.

По-друге

Розподілена система керування базою даних (СКБД) – це програмний комплекс, призначений для управління розподіленими базами даних і забезпечує прозорий доступ користувачів до розподіленої інформації.

Прозорий доступ означає непомітний для користувача, тобто у користувача має складатися враження ніби він працює з єдиною базою даних, яка розміщена на його власному комп'ютері.

Розподілена СКБД, порівняно з централізованою СКБД, повинна мати додатковий набір функціональних можливостей.

Одним із різновидів розподіленої СКБД є мультибазова система – розподілена система управління базами даних, в якій управління кожним вузлом здійснюється автономно.

По-третє

Розподілена обробка даних (distributed processing) використовує централізовану базу даних, доступ до якої може здійснюватись з різних комп'ютерів мережі. Вузол, на якому розміщена база даних, називають сервером бази даних. Ключовим моментом у визначенні розподіленої СКБД є твердження, що система працює з даними, які фізично розподілені в мережі, тобто фрагменти розподіленої бази даних розміщені на різних вузлах.

По-четверте

За способом розміщення розподілені бази даних ділять на зосереджені і розо-середжені.

Фундаментальний принцип створення розподілених баз даних («правило 0») передбачає їх організацію таким чином, щоб для користувача розподілені системи виглядали так само, як і нерозподілені.

Література

- 1 Трофименко О. Г. Організація баз даних : Навч. посібник . [Текст] / О. Г. Трофименко, Ю. В. Прокоп, Н. І. Логінова, І. М. Копитчук. 2-ге вид. виправ. і доповн.– Одеса : Фенікс, 2019. – 246 с.
- 2 Харів Н. О. Бази даних та інформаційні системи: навчальний посібник . [Текст] / Н. О. Харів. – Рівне : НУВГП, 2018. – 127 с.
- 3 Ярцев В.П. Організація баз даних та знань: навчальний посібник. [Текст] / В.П. Ярцев – К. ДУТ 2018.-214с.

Навчальні матеріали до самостійного вивчення теми

1 Концепція розподілених баз даних.

1.1 Основні поняття

Основною передумовою розробки систем, що використовують бази даних, є прагнення об'єднати всі дані, які обробляються на підприємстві, в єдине ціле та забезпечити контрольований доступ до них. Сучасні потужні підприємства часто мають велику кількість підрозділів, які можуть фізично розташовуватись за сотні та навіть тисячі кілометрів один від одного. Кожний з цих підрозділів має свою локальну базу даних. Якщо ці бази мають різні архітектури та використовують різні протоколи зв'язку, то їх розглядають як інформаційні острови, важкодоступні для інших. В такому випадку виникає необхідність об'єднати розрізнені бази даних в одне логічне ціле, тобто створити розподілену базу даних.

Розподілена база даних – це сукупність логічно зв'язаних баз даних або частин однієї бази, які розпаралелені між декількома територіально-розподіленими ПЕОМ і забезпечені відповідними можливостями для управління цими базами або їх частинами. Тобто, розподілена база даних реалізується на різних просторово розосереджених обчислювальних засобах, разом з організаційними, технічними і програмними засобами її створення і ведення.

В дійсності розподілена база даних є віртуальною базою даних, компоненти якої фізично зберігаються на декількох різних реальних базах даних на декількох різних вузлах. На рисунку 1 наведено приклад типової топології розподіленої бази даних.

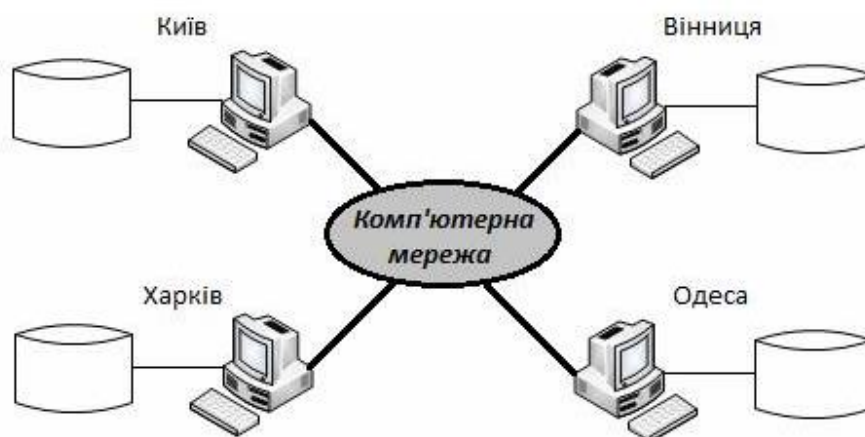


Рисунок 1 – Приклад типової топології розподіленої бази даних

Використовувати розподілені бази даних ефективно і доцільно в предметних областях, які характеризуються:

- занадто великими обсягами даних, які зберігаються і обробляються;
- фізичною розосередженістю місць збирання, зберігання і використання даних;
- наявністю розвинутих засобів обчислювальної техніки і мереж передачі даних;
- можливістю обробки більшої частини інформації в місцях, де вона виникає або зберігається;
- необхідністю одночасного виконання масової обробки інформації тощо.

Розподілена система керування базою даних (СКБД) – це програмний комплекс, призначений для управління розподіленими базами даних і забезпечує прозорий доступ користувачів до розподіленої інформації.

Прозорий доступ означає непомітний для користувача, тобто у користувача має складатися враження ніби він працює з єдиною базою даних, яка розміщена на його власному комп'ютері.

Розподілена СКБД, порівняно з централізованою СКБД, повинна мати додатковий набір функціональних можливостей:

- розширені служби встановлення з'єднань повинні забезпечувати доступ до віддалених вузлів і дозволяти передавати запити і дані між вузлами, які входять у мережу;
- розширені засоби ведення каталогу, що дозволяють зберігати відомості про розподіл даних в мережі;
- засоби обробки розподілених запитів, включаючи механізми оптимізації запитів і організації віддаленого доступу до даних;
- розширені функції управління захистом, що дозволяють забезпечити дотримання правил авторизації та прав доступ до розподілених даних;
- розширені функції управління паралельним виконанням, які дозволяють підтримувати цілісність копіювання;
- розширені функції відновлення, що враховують ймовірність відмов в роботі окремих вузлів і відмов ліній зв'язку.

Всі функції розподіленої СКБД мають виконуватись прозоро для користувача.

До складу розподіленої СКБД повинні входити такі компоненти:

- комп'ютерні робочі станції (сайти або вузли), що формують мережеву систему;
 - компоненти мережевого обладнання та програмного забезпечення кожної робочої станції, які дозволяють усім вузлам взаємодіяти один з одним та обмінюватися даними;
 - комунікаційні пристрої, які переносять дані з однієї робочої станції на іншу;
 - процесор транзакцій (transaction processor, TP) – програмний компонент, що знаходиться на кожному комп'ютері, де виконується запит даних. Процесор транзакцій отримує і обробляє запити (віддалені та локальні);
 - процесор даних (data processor, DP) – програмний компонент, розташований на кожному комп'ютері, де зберігаються і обробляються дані, розташовані на даному вузлі. Процесор даних може навіть являти собою централізовану СКБД.
- На рисунку 2 показано розміщення і взаємодію усіх компонентів розподіленої СКБД.

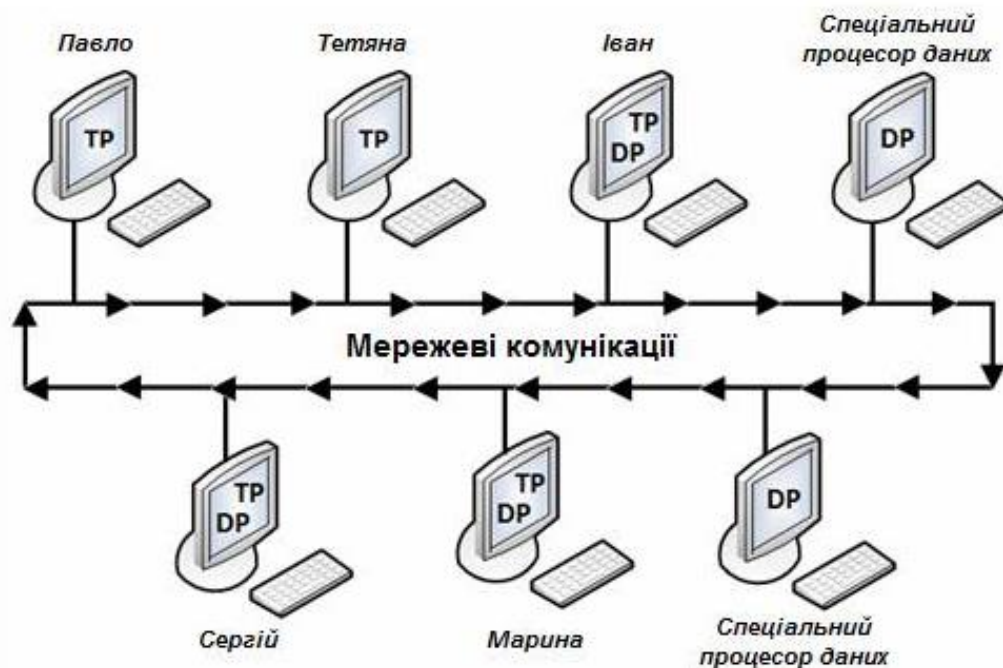


Рисунок 2 – Взаємодія компонентів системи розподіленої БД

Кожний TP може отримати доступ до даних на будь-якому DP і кожний DP обробляє всі запити локальних даних від будь-якого TP.

Зв'язок між TP і DP здійснюється відповідно до специфічних правил або протоколів, які визначають, як система розподіленої бази даних:

- організовує інтерфейс з мережею для передачі даних і команд між процесорами даних (DP) і процесорами транзакцій (TP);

- синхронізує всі дані, отримані від DP, і передає отримані дані на відповідні TP;
- забезпечує функції загального управління БД в розподіленій системі (безпека, управління паралельним виконанням, створення резервних копій і відновлення).

Процесори TP і DP можуть бути розміщені на одному і тому ж комп'ютері, дозволяючи користувачам отримувати доступ до локальних і віддалених даних, не турбуючись про їх місцезнаходження. Теоретично DP може представляти собою незалежну централізовану СКБД з відповідним інтерфейсом для підтримки віддаленого доступу до інших незалежних СКБД в мережі.

Одним із різновидів розподіленої СКБД є *мультибазова система* – розподілена система управління базами даних, в якій управління кожним вузлом здійснюється автономно.

Мультибазові системи дозволяють кінцевим користувачам різних вузлів отримувати доступ і спільно використовувати дані без необхідності фізичної інтеграції існуючих баз даних. Вони забезпечують користувачам можливість управляти базами даних їх власних вузлів без будь-якого централізованого контролю, який обов'язково присутній у звичайних типах розподілених СКБД. Адміністратор локальної бази даних може дозволити доступ до певної частини своєї бази даних за допомогою створення схеми експорту, яка визначає, до яких елементів локальної бази даних зможуть отримувати доступ зовнішні користувачі.

Існують так звані *необ'єднані* (не мають локальних користувачів) і *об'єднані* мультибазові системи. Об'єднана система являє собою деякий гібрид розподіленої та централізованої систем, оскільки вона виглядає як розподілена система для віддалених користувачів і як централізована система – для локальних.

Іншими словами, мультибазова СКБД непомітно для користувача розміщується над існуючими системами баз даних і файловими системами і розглядається користувачами як єдина база даних. Мультибазова СКБД підтримує глобальну схему, на основі якої користувачі можуть формувати запити і модифікувати дані. Мультибазова СКБД працює тільки з глобальною схемою, тоді як локальні СКБД власними силами забезпечують підтримку даних всіх своїх користувачів. Глобальна схема створюється шляхом

об'єднання схем локальних баз даних. Програмне забезпечення мультибазової СКБД попередньо транслює глобальні запити і перетворює їх на запити та оператори модифікації даних відповідних локальних СКБД. Потім отримані після виконання локальних запитів результати зливаються в єдиний глобальний результуючий набір, що надається користувачеві. Крім того, мультибазова СКБД здійснює контроль за виконанням фіксації або відкату окремих операцій глобальних транзакцій локальних СКБД, а також забезпечує збереження цілісності даних в кожній з локальних баз даних. Програми мультибазової СКБД управляють різними шлюзами, за допомогою яких вони контролюють роботу локальних СКБД.

Одним із прикладів мультибазової системи є система UniSQL компанії Cincom Corporation. Вона дозволяє розробляти програми за допомогою єдиного глобального уявлення і єдиної мови доступу до бази даних для роботи з багатьма різнорідними реляційними і об'єктно-орієнтованими СКБД.

1.2 Відмінності між розподіленими системами баз даних, засобами розподіленої обробки даних та паралельними системами баз даних

Дуже важливо розуміти відмінність між розподіленими системами баз даних і засобами розподіленої обробки даних.

Розподілена обробка даних (distributed processing) використовує централізовану базу даних, доступ до якої може здійснюватись з різних комп'ютерів мережі. Вузол, на якому розміщена база даних, називають сервером бази даних. Ключовим моментом у визначенні розподіленої СКБД є твердження, що система працює з даними, які фізично розподілені в мережі, тобто фрагменти розподіленої бази даних розміщені на різних вузлах. Топологію системи розподіленої обробки даних зображено на рисунку 3. Як видно з рисунка 3, база даних розміщена лише на одному вузлі, в той час як на рисунку 1 можна побачити, що в розподіленій СКБД кожен вузол може містити свій фрагмент розподіленої бази даних.

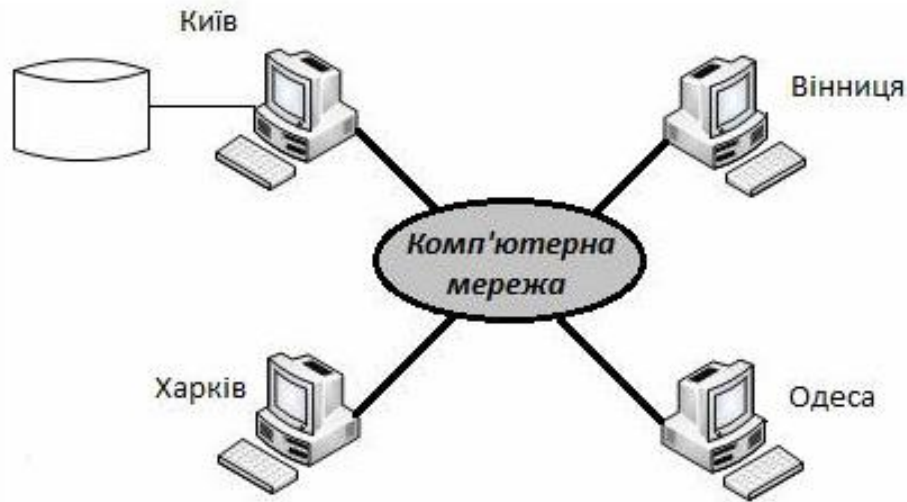


Рисунок 3 – Топологія середовища розподіленої обробки даних

При розподіленій обробці даних логічні процеси бази даних розподіляються серед двох або більше фізично незалежних вузлів, об'єднаних в мережу. Наприклад, розподілена обробка може виконувати введення/виведення даних, вибірку даних і перевірку даних на одному комп'ютері, а на іншому комп'ютері випускати звіт на основі отриманої інформації. Кожен вузол може мати доступ до даних та оновлювати базу даних.

Розподілена обробка даних не вимагає розподіленої бази даних, але розподілена база даних обов'язково вимагає розподіленої обробки інформації.

Спільною рисою розподіленої обробки даних і розподіленої бази даних є необхідність в локальній мережі, яка зв'язує всі компоненти між собою.

Також варто розуміти відмінності, які існують між розподіленими і паралельними СКБД. Паралельна СКБД функціонує з використанням декількох процесорів і жорстких дисків, що дозволяє їй розпаралелити виконання деяких операцій з метою підвищення загальної продуктивності обробки.

Застосування паралельних СКБД дозволяє об'єднати декілька малопотужних машин для отримання такого ж рівня продуктивності, що й у випадку однієї більш потужної машини, але при цьому досягається вищий рівень масштабованості та надійності системи в порівнянні з однопроцесорною СКБД.

Для надання декільком процесорам спільного доступу до однієї і тієї ж бази даних паралельна СКБД повинна забезпечувати управління сумісним доступом до ресурсів. Залежно від того, які саме ресурси поділяються, розрізняють три види паралельних СКБД:

- системи з поділом пам'яті;
- системи з поділом дисків;
- системи без поділу обчислювальних ресурсів.

Паралельна *система без поділу обчислювальних ресурсів* іноді розглядається як розподілена СКБД, однак в такій системі розподіл даних зумовлений лише прагненням до підвищення продуктивності. Крім того, вузли розподіленої СКБД зазвичай розділені географічно, знаходяться під управлінням різних адміністраторів і з'єднані між собою відносно повільними мережевими з'єднаннями, тоді як вузли паралельної СКБД найчастіше розташовуються на одному і тому ж комп'ютері або в межах одного і того ж виробничого майданчика. Архітектура системи баз даних з паралельною обробкою зображена без поділу обчислювальних ресурсів на рисунку 4.

Системи з поділом пам'яті складаються з тісно пов'язаних між собою компонентів, до числа яких входить кілька процесорів, які поділяють загальну системну пам'ять. Ця архітектура, яку ще називають архітектурою з симетричною багатопроцесорною обробкою (SMP), на даний час отримала широке поширення і застосовується для найрізноманітніших обчислювальних платформ, від персональних робочих станцій, що містять кілька паралельно працюючих мікропроцесорів, великих RISC-систем і до найбільших мейнфреймів. Ця архітектура забезпечує швидкий доступ до даних для обмеженого набору процесорів, кількість яких звичайно не перевершує 64. В іншому випадку взаємодія по мережі стає вузьким місцем всієї системи.

Системи з поділом дисків створюються з менш тісно пов'язаних між собою компонентів. Вони є оптимальним варіантом для додатків, які успадкували високу централізацію обробки і повинні забезпечувати найвищі показники доступності та продуктивності. Кожен з процесорів має безпосередній доступ до всіх спільно використовуваних дискових пристроїв, але має власну оперативну пам'ять. Як і у випадку архітектури без поділу обчислювальних ресурсів, архітектура з поділом дисків виключає вузькі місця, пов'язані з спільно використовуваною пам'яттю. Однак, на відміну від архітектури без поділу обчислювальних ресурсів, дана архітектура виключає згадані вузькі місця без внесення додаткових витрат, пов'язаних з фізичним розподілом даних по окремим пристроям. Колективні дискові системи іноді називають кластерами.



Рисунок 4 – Архітектура системи баз даних з паралельною обробкою без поділу обчислювальних ресурсів

Системи без поділу обчислювальних ресурсів (цю архітектуру інакше називають архітектурою з масовою паралельною обробкою) використовують схему, в якій кожен процесор, який є частиною системи, має свою власну оперативну і дискову пам'ять. База даних розподілена між всіма дисковими пристроями, які підключені до окремих, пов'язаних з цією базою даних обчислювальних підсистем, в результаті чого всі дані прозора доступні користувачам кожної з цих підсистем. Така архітектура забезпечує більш високий рівень масштабованості, ніж системи з поділом пам'яті, і дозволяє легко підтримувати велику кількість процесорів. Однак оптимальна продуктивність буде досягнута тільки в тому випадку, якщо необхідні дані зберігаються локально.

Паралельні технології зазвичай використовуються у випадку виключно великих баз даних, розміри яких можуть досягати декількох терабайт (10^{12} байт), або в системах, які забезпечують виконання тисяч транзакцій в секунду. Подібні системи потребують доступу до великого обсягу даних і повинні забезпечувати прийнятний час реакції на запит.

1.3 Класифікація розподілених баз даних

За способом розміщення розподілені бази даних ділять на зосереджені і розосереджені.

Зосереджені (або централізовані) розподілені бази даних фізично розміщені в одному місці. Для обміну інформацією між окремими (локальними) підбазами використовуються канали зв'язку прямого доступу. Обмін даними між взаємозв'язаними підбазами здійснюється без помітних обмежень на обсяги і характер інформації, що передається. Такі бази даних мають ряд переваг:

- просту побудову бази даних;
- зведення до мінімуму дублювання інформації;
- максимальну уніфікацію методів зберігання, коригування і пошуку інформації.

Проте зосереджені бази даних в одному місці – вузлі мережі – мають цілий ряд недоліків: при централізації зберігання значно збільшується час на передачу інформації і за рахунок цього зростає час реакції системи; централізована система обмежена обсягами пам'яті ЕОМ тощо.

Розосереджені (або децентралізовані) розподілені бази даних фізично розміщені в різних місцях – вузлах обчислювальної мережі. Обмін інформацією між підбазами здійснюється з використанням каналів зв'язку. Як підбази розподіленої бази даних можуть використовуватись зосереджені (централізовані) бази даних і окремі (локальні) підбази.

Обмін інформацією між взаємозв'язаними підбазами здійснюється головним чином результатною (обробленою, узагальненою) інформацією. При виконанні запиту в таких системах використовується декомпозиція запиту на підзапити до локальних підбаз і паралельне виконання виділених підзапитів у різних вузлах обчислювальної мережі. Ці бази даних мають безперечні переваги у порівнянні з централізованими:

- обсяги пам'яті обмежені пам'яттю не однієї ЕОМ, а сумарною пам'яттю ЕОМ, які знаходяться в усіх вузлах мережі;
- зменшуються затрати на передавання інформації, так як у кожному вузлі знаходиться та інформація, яка необхідна конкретному користувачу і по можливості забезпечує всі його інформаційні потреби.

Однак розосереджена база даних призводить до неминучого дублювання деякої інформації, безконтрольності її зростання, а також значно ускладнюється проблема зберігання несуперечності інформації.

Залежно від рівня підтримки різних типів локальних СКБД, розподілені СКБД поділяються на гомогенні (однорідні) та гетерогенні (різнорідні).

В гомогенних системах усі вузли використовують один і той же тип СКБД. У різнорідних системах на вузлах можуть функціонувати різні типи СКБД, що використовують різні моделі даних, тобто різнорідна система може включати вузли з реляційними, мережевими, ієрархічними або об'єктно-орієнтованими СКБД.

Однорідні системи значно простіше проектувати і супроводжувати. Крім того, подібний підхід дозволяє поетапно нарощувати розміри системи, послідовно додаючи нові вузли до вже існуючої розподіленої системи. Додатково з'являється можливість підвищувати продуктивність системи за рахунок організації на різних вузлах паралельної обробки інформації.

Різнорідні системи зазвичай виникають в тих випадках, коли незалежні вузли, які вже експлуатують свої власні системи з базами даних, з часом інтегруються у новостворювану розподілену систему. У різнорідних системах для організації взаємодії між різними типами СКБД потрібно забезпечити перетворення переданих повідомлень. Для забезпечення прозорості щодо типу використовуваної СКБД користувачі кожного з вузлів повинні мати можливість формувати запити мовою тієї СКБД, яка використовується на їх локальному вузлі. Система повинна взяти на себе пошук необхідних даних і виконання всіх необхідних перетворень переданих повідомлень. У загальному випадку дані можуть бути затребувані з іншого вузла, який характеризується такими особливостями:

- інший тип використовуваного обладнання;
- інший тип використовуваної СКБД;
- інший тип застосовуваних обладнання та СКБД.

Якщо використовується інший тип обладнання, але на вузлах застосовуються однакові СКБД, методи виконання перетворень цілком очевидні і включають заміну кодів і зміну довжини машинного слова. Якщо типи використовуваних на вузлах СКБД різні, процедура перетворення ускладнюється тим, що необхідно перетворювати структури даних однієї моделі даних в еквівалентні структури даних іншої моделі даних. Наприклад, відношення в реляційній моделі даних повинні бути перетворені в записи і набори, характерні для мережевої моделі даних. Крім того, доводиться транслювати текст запи-

тів з однієї мови в іншу (наприклад, запити з оператором SELECT мови SQL може знадобитися перетворити в запити з операторами GET і FIND мови маніпулювання даними мережевої СКБД). Якщо відрізняються і тип використовуваного обладнання, і тип програмного забезпечення, то необхідно виконати обидва види трансляції. Все викладене вище надзвичайно ускладнює обробку даних в різномірних розподілених СКБД.

В гетерогенних системах може виникати проблема семантичної неоднорідності. Наприклад, атрибути, що мають у різних схемах одне і те ж ім'я, фактично можуть представляти абсолютно різні поняття. Аналогічно, атрибути з різними іменами фактично можуть представляти одну і ту ж характеристику.

Типове рішення, що застосовується в деяких реляційних системах, полягає в тому, що окремі частини різномірних розподілених систем повинні використовувати шлюзи, призначені для перетворення мови та моделі даних кожного з використовуваних типів СКБД в мову і модель даних реляційної системи. Однак підходу із застосуванням шлюзів властиві деякі серйозні обмеження. По-перше, шлюзи не дозволяють організувати систему управління транзакціями навіть для окремих пар систем. Іншими словами, шлюз між двома системами представляє собою не більш ніж транслятор запитів. Також шлюзи не дозволяють системі координувати управління паралельним виконанням та процедурами відновлення транзакцій, які включають оновлення даних в обох базах. По-друге, використання шлюзів дозволяє вирішити лише завдання трансляції запитів з мови однієї СКБД на мову іншої. Тому вони, як правило, не дозволяють вирішити проблему створення однорідної структури і усунути відмінності між представленнями даних у різних схемах.

Для подолання проблем гетерогенних систем комітет Open Group організував робочу групу (Specification Working Group – SWG), покликану підготувати специфікації, що регламентують інфраструктуру середовища бази даних, яка включає такі елементи:

- уніфікований і досить потужний інтерфейс мови SQL (SQL API), який дозволяє створювати клієнтські програми таким чином, щоб вони не були прив'язані до конкретного типу використовуваної СКБД;
- уніфікований протокол доступу до бази даних, що дозволяє безпосередньо взаємодіяти СКБД різних типів без необхідності використання будь-якого шлюзу;

- уніфікований мережний протокол, що дозволяє здійснювати взаємодію СКБД різних типів.

Найважливішим завданням цієї групи слід вважати пошук способу, який дозволяє в одній транзакції виконувати обробку даних, що містяться в декількох базах, керованих СКБД різних типів, причому без необхідності застосування будь-яких шлюзів.

1.4 Переваги та недоліки систем керування розподіленими базами даних

Система керування розподіленою БД порівняно з традиційними системами має ряд переваг:

- дані розташовуються близько до найзатребуванішого вузла. Дані в розподіленій БД розміщуються на тому вузлі, де в них є найбільша потреба;
- високий ступінь локальної автономності. Користувачі, які найчастіше працюють з даними на деякому локальному вузлі можуть здійснювати локальний контроль над необхідними їм даними, встановлювати або регулювати локальне обмеження на їх використання. Адміністратор глобальної бази даних (АБД) відповідає за систему в цілому. Як правило, частина цієї відповідальності делегується на локальний рівень, завдяки чому АБД локального рівня отримує можливість управляти локальною СКБД;
- швидкий доступ до даних. Кінцеві користувачі часто працюють тільки з деякою підмножиною даних компанії. При цьому підмножина може зберігатися локально, і система бази даних забезпечить більш оперативний доступ до даних, ніж централізована БД, де дані зберігаються віддалено;
- підвищення продуктивності. В розподіленій БД має місце розпаралелення процесів обробки даних. Оскільки кожен вузол працює лише з частиною бази даних, то ступінь використання центрального процесора та служб введення-виведення може виявитися нижчим, ніж у випадку централізованої БД;
- підвищення надійності та доступності даних. У централізованих СКБД відмова центрального комп'ютера приведе до припинення функціонування всієї СКБД. На відміну від них, розподілена СКБД дозволяє перемістити операції з вузла, що вийшов з ладу, на інший. Навантаження системи розподіляється між іншими робочими станціями, оскільки, завдяки реплікації, дані зберігаються на декількох вузлах;

- модульність системи. Розширення розподіленої бази даних відбувається шляхом додавання до мережі нового вузла, який не впливає на функціонування вже існуючих вузлів. В централізованих СКБД розширення бази даних може вимагати заміни обладнання та програмного забезпечення;

- зменшення вартості організації і витрат на експлуатацію баз даних. У 1960-ті роки потужність обчислювальних засобів зростала пропорційно квадрату вартості її устаткування, тому система, вартість якої була втричі вища за вартість даної, перевершувала її за потужністю у дев'ять разів. Ця залежність отримала назву закону Гроша. Однак сьогодні вважається загальноприйнятим положення, згідно з яким набагато дешевше зібрати з невеликих комп'ютерів систему, потужність якої буде еквівалента потужності одного великого комп'ютера (мейнфрейма). Виявляється, що значно вигідніше встановлювати в підрозділах організації власні малопотужні комп'ютери та додавати в мережу нові робочі станції, ніж модернізувати систему з мейнфреймом. В багатьох корпораціях на мейнфреймах виконують лише вузькоспеціалізовані задачі, а всі інші – на персональних комп'ютерах;

- зменшення вартості передачі даних. Вартість передачі даних через комп'ютерну мережу є відносно високою порівняно з локальним доступом до даних. Вона визначається не лише кількістю переданих бітів інформації, а й витратами на підтримку багаторівневого мережевого протоколу, який управляє підготовкою інформаційних пакетів до передачі і збором отриманих пакетів на вузлі-приймачі. Реалізація кожного рівня протоколу вимагає значних обчислювальних витрат. У розподіленій СКБД забезпечується можливість розмістити дані на тому вузлі, де в них є найбільша потреба і обробляти їх локально;

- незалежність від вузла обробки даних. Кінцевий користувач отримує доступ до будь-якої наявної копії даних, а запит кінцевого користувача обробляється будь-яким вузлом в місці розташування даних. Іншими словами, запит не залежить від конкретного вузла обробки даних: будь-який доступний вузол може обробити запит користувача;

- збільшення обсягу збережених і доступних для обробки даних. Обсяг даних не обмежується об'ємом пам'яті мейнфрейма. Перевантаження через збільшення розміру бази даних зазвичай усуваються шляхом додавання до мережі нових обчислювальних потужностей і пристроїв зовнішньої пам'яті;

- зменшення обсягів даних, які пересилаються. Досягається завдяки тому, що частина даних зберігається і обробляється локально.

На жаль, сучасні розподілені СКБД мають ряд недоліків:

- складність управління і контролю. Управління розподіленими даними – більш важке завдання, ніж управління централізованими даними. Додатки повинні визначати місцезнаходження даних і вміти потім зв'язувати в єдине ціле інформацію, отриману з різних місць. Адміністратори БД повинні мати можливість координувати дії бази даних, щоб запобігти погіршенню якості БД через аномалії даних. Особливу увагу слід приділяти управлінню транзакціями, паралельному виконанню, безпеці, резервному копіюванню, реплікації даних, відновленню процесів, оптимізації запитів, вибору шляхів доступу і т. д;

- ускладнення процедури розробки розподіленої БД. Розробка розподілених баз даних, вимагає прийняття рішення про фрагментацію даних, розподіл фрагментів по окремим вузлам і реплікацію даних;

- безпека. В централізованих системах доступ до даних легко контролюється. В розподілених системах відповідальність за управління даними розподіляється між різними людьми, що знаходяться в різних місцях, а локальні мережі все ще залишаються уразливими у відношенні безпеки;

- недостатня стандартизація. Незважаючи на те, що розподілені бази даних залежать від ефективності комунікацій, поки не існує стандартного протоколу обміну інформацією на рівні баз даних (хоча TCP/IP є фактично стандартним протоколом на рівні мереж, відсутній стандарт протоколу на рівні додатків). Відсутність стандартів істотно обмежує потенційні можливості розподілених СКБД. Крім того, не існує інструментальних засобів і методологій, здатних допомогти користувачам перетворити централізовані системи в розподілені;

- складність управління середовищем даних. Організувати доступ до диска і зберігання даних у розподіленому середовищі складніше, ніж в централізованій системі, тому управління таким середовищем даних стає більш складним у відношенні як людських ресурсів, так і програмного забезпечення;

- підвищення вартості навчання. Вартість навчання в розподіленій моделі, як правило, вище, ніж в централізованій, і часто порівнюється з вартістю обладнання;

– нестача досвіду. Ще не накопичено достатньо досвіду промислової експлуатації розподілених систем. Таке становище є значним стримуючим фактором для потенційних прихильників розподіленої технології;

– підвищені вимоги до умов зберігання даних. У розподілених БД декілька копій даних необхідно розміщувати на декількох сайтах, що вимагає додаткових ресурсів (місце на жорсткому диску). Цей недолік не є дуже істотним, оскільки вартість зберігання інформації на жорстких дисках швидко зменшується.

Сьогодні розподілені БД з успіхом експлуатуються, однак гнучкість і потужність, якими вони теоретично володіють, використовується не в повній мірі. Властива розподіленим базам даних складність вимагає невідкладної розробки стандартів на протоколи управління транзакціями, паралельного виконання, безпеки, резервного копіювання і відновлення, оптимізації, вибору шляхів доступу і т. д.

1.5 Принципи створення розподілених БД

Фундаментальний принцип створення розподілених баз даних («правило 0») передбачає їх організацію таким чином, щоб для користувача розподілені системи виглядали так само, як і нерозподілені.

З фундаментального принципу витікають додаткові правила, запропоновані К.Дейтом:

1. Незалежність локального вузла. Кожний локальний вузол може працювати як незалежна, або автономна СКБД. Всі операції на вузлі контролюються цим вузлом: безпека, управління паралельним виконанням, резервне копіювання і відновлення даних.

2. Незалежність від центрального вузла. Всі вузли мають рівні можливості. Жодний вузол в мережі не повинен звертатись до «центрального» вузла з метою отримання певного централізованого сервісу.

3. Незалежність від збоїв. Функціонування системи не залежить від збою на якомусь вузлі. Система продовжує виконання операцій навіть при несправності вузла або при розширенні мережі.

4. Прозорість місця розташування. Користувачу не потрібно знати фізичне місце розташування даних, щоб здійснити їх пошук. Користувач працює з даними так, ніби вони розташовані на його локальному вузлі.

5. Прозорість фрагментації. Користувачу не потрібно знати імена фрагментів БД, щоб отримати доступ до них. Користувач бачить єдину логічну базу даних.

6. Прозорість реплікації. Для користувачів повинно бути створено таке середовище, щоб вони могли вважати, що в дійсності дані не дублюються. Тобто, користувачу не потрібно мати відомості про існуючі копії фрагментів.

7. Розподілена обробка запитів. Для обробки запиту може знадобитись звернення до декількох вузлів. В розподіленій системі може існувати багато способів пересилання даних для виконання цього запиту. СУРБД повинна виконати оптимізацію запиту прозоро для користувача.

8. Управління розподіленими транзакціями. Транзакції можуть оновлювати дані на декількох різних вузлах. Виконання транзакцій відбувається прозоро для користувача.

9. Апаратна незалежність. Система повинна виконуватись на різних апаратних платформах.

10. Незалежність від операційної системи. Система повинна виконуватись в будь-якій операційній системі.

11. Незалежність від мережі. Можливість підтримувати багато типів комунікаційних мереж.

12. Незалежність від типу бази даних. Необхідно, щоб екземпляри БД на різних вузлах всі разом підтримували один і той же інтерфейс, і зовсім необов'язково, щоб це були копії однієї і тієї ж версії БД. Іншими словами, розподілені бази даних можуть бути неоднорідними.

На сьогодні жодна з розподілених баз даних не відповідає усім наведеним правилам, однак правила Дейта повністю описують розподілені бази даних і відіграють важливу роль при їх розробці.

Питання та завдання для самоконтролю знань студентів

1. Що називають розподіленою базою даних? Яка принципова відмінність розподіленої бази даних від централізованої?

2. Які основні функції повинна виконувати система керування розподіленою базою даних?

3. Чим відрізняється розподілена база даних від середовища розподіленої обробки даних?
4. За яким принципом відбувається класифікація розподілених баз даних?
5. Дайте стислу характеристику зосереджених (централізованих) розподілених баз даних.
6. Дайте стислу характеристику розсереджених (децентралізованих) розподілених баз даних.
7. За яким принципом відбувається класифікація розподілених СКДБ ?
8. Дайте стислу характеристику гомогенних (однорідних) розподілених СКДБ.
9. Дайте стислу характеристику гетерогенних (різномірних) розподілених СКДБ.
10. Виділіть основні переваги систем керування розподіленими базами даних.
11. Виділіть основні недоліки систем керування розподіленими базами даних.
12. Виділіть основний принцип створення розподілених БД.