МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені В.Н. КАРАЗІНА

Кафедра міжнародних відносин, міжнародної інформації та безпеки

КОМПЛЕКС НАВЧАЛЬНО-МЕТОДИЧНОГО ЗАБЕЗПЕЧЕННЯ

з дисципліни
Основи організації бази даних

рівень вищої освіти перший (бакалаврський)

галузь знань 29 «Міжнародні відносини»

спеціальність 291 «Міжнародні відносини, суспільні комунікації та регіональні студії»

освітня програма «Міжнародна інформація та міжнародні комунікації»

спеціалізація _____

вид дисципліни за вибором

факультет Міжнародних економічних відносин та туристичного бізнесу

Укладач: к.е.н., доц. Чала О.В.

1. Навчальний контент:

Програма навчальної дисципліни «Основи організації бази даних» складена відповідно до освітньо-професійної програми підготовки бакалавра з «Міжнародної інформації та міжнародних комунікацій» спеціальності 291 «Міжнародні відносини, суспільні комунікації та регіональні студії»

даних

Лекція 1.

Місце СУБД в системах обробки інформації

1.1. Системи управління базами даних.
1.2. Функціональні характеристики СУБД.
1.3. Основні вимоги до СУБД.

1.1. Системи управління базами даних.

Системи управління базами даних (СКБД) - це програмні комплекси,
Призначені для роботи зі спеціально організованими файлами (масивами даних,довготривало збереженими в зовнішній пам'яті обчислювальних систем), які називаються *базами даних.* Основними функціями СУБД є управління файлами БД («внутрішня» функція) і обробка прикладних програм (запитів) користувачів БД («зовнішня» функція). До забезпечує функцій СУБД відносяться: підтримка цілісності БД в процесі її експлуатації, захист БД від несанкціонованого доступу, управління обміном даними між БД і зовнішнім середовищем (в т.ч. управління розподіленою обробкою даних).
Крім того, сучасні СУБД часто оснащуються додатковими інструментальними засобами розробки додатків (прикладних задач обробки даних). СУБД як програмний продукт характеризується великою трудомісткістю виготовлення, високою наукоемкостью, тривалими термінами розробки, значною вартістю. В даний час на ринку інформаційних технологій пропонуються десятки різних СУБД, доведених до «комерційного» (тобто рекомендованого до промислової експлуатації) зразка. відповідно, перед споживачем, для тих, до створення інформаційної системи, постає проблема усвідомленого вибору необхідної йому моделі, версії і конфігурації СУБД.
Сучасні БД служать для збереження і обробки інформації, що стосується самих різних аспектів діяльності як окремих підрозділів (що складаються може бути всього з кількох людей), так і великих організацій і підприємств (де активними користувачами можуть бути сотні і тисячі чоловік). При виборі типу і конфігурації СУБД прийнято виділяти такі класи інформаційних систем, що функціонують на основі технології БД (*систем баз даних):*
- системи оперативної обробки транзакцій (OLTP). Характеризуються тим, що до БД надходить інтенсивний потік запитів на оновлення даних, в той час як потік запитів на додавання великих обсягів даних невеликий. Типовим прикладом є системи резервування квитків на пасажирському транспорті;
- системи підтримки прийняття рішень Характеризуються використанням БД великого об'єму, в яких накопичується інформація за тривалий період часу і здійснюються в основному операції пошуку і зчитування даних, що використовуються додатками для формування звітів. Типовим прикладом можуть служити системи верхнього рівня управління підприємством;
- системи пакетної обробки (РР). характеризуються здатністю тривалий час працювати в автономному режимі з мінімальною участю оператора. Як правило, БД тут використовується обмеженим числом прикладних програм. Типовий приклад - системи управління технологічними процесами.

1.2. Функціональні характеристики СУБД.

Основні можливості СУБД визначаються набором таких її функціонально-технологічних характеристик, як підтримувана модель даних, рівень вхідного мови, масштабованість і переносимість, архітектура розподіленої обробки.

Модель даних. Визначає способи та форми подання даних на зовнішньому (призначеному для користувача) рівні. Слід підкреслити, що сучасні СУБД орієнтовані в основному на обробку так званої фактографічної інформації, подається впорядкованою сукупністю фактів (значень у вигляді чисел, рядків символів, логічних констант і т.п.). Ці значення, як правило, об'єднуються в порівняно невеликі послідовності - записи, які відповідають окремим об'єктам (явищ) прикладної предметної області. Сукупності однорідних записів об'єднуються в таблиці (така форма подання інформації, як відомо, найбільш широко застосовується як в «ручних», так і автоматизованих інформаційних системах). Таким чином, модель даних задає правила структуризації таблиць, їх логічного зв'язування в БД, а також правила виконання операцій над елементами «логічної» структури даних. Незважаючи на те, що існує безліч моделей представлення табличних даних, в даний час домінуюче становище на ринку комерційних СУБД займають *реляційні* системи, тобто СУБД, що підтримують реляційну модель даних. (Ця модель являє БД як сукупність жорстко не пов'язаних між собою таблиць, а операції тут мають простий і точний зміст, наприклад: «додати рядок в таблицю», «видалити з таблиці рядки, що відповідають деякому логічному умові», «об'єднати (злити) дві однорідні таблиці в одну »і т.п.). Крім простоти маніпулювання з даними реляційні системи мають ряд інших важливих переваг над нереляціоннимі.

У подальшому обговоренні в силу зазначених обставин ми зосередимося виключно на реляційних СУБД. Слід підкреслити, що сучасні реляційні СУБД надають користувачеві можливість працювати не тільки з фактографічної інформацією, але і з інформацією, представленою в інших формах (великі текстові документи, графіки, малюнки і т.д.). У цьому випадку окремі поля реляційних таблиць замість традиційних значень містять посилання на нетипові для СУБД інформаційні об'єкти, а сама СУБД оснащена засобами доступу до цих об'єктів (і, можливо, обмеженими засобами їх спеціалізованої обробки).

Підтримка моделі даних полягає в реалізації засобами СУБД мови обробки даних, що відповідає цій моделі. У загальному випадку СУБД можуть оснащуватися декількома вхідними мовами, відмінними рівнем деталізації команд на обробку даних. Мови дуже високого рівня реалізують обмежений набір операцій реляційної моделі і орієнтовані на кінцевого користувача. Вони дозволяють з найменшими затратами праці формулювати прості запити на обробку даних (набір яких, втім, може виявитися достатнім для вирішення всіх необхідних користувачеві завдань).

Мови, що реалізують операції реляційної моделі в повному обсязі, також характеризуються малим ступенем деталізації виконуваних команд і відносяться до мов обробки даних високого рівня. В даний час в індустрії систем обробки даних склався стандарт реляційного мови високого рівня, званий *непроцедурного* SQL, і підтримуваний більшістю сучасних СУБД (як правило, з деякими відхиленнями від стандарту). Непроцедурного SQL є більш, ніж реляційно повним мовою, оскільки включає ряд додаткових операцій над даними (арифметичні операції, операції порівняння подстрок символів та ін.). Проте, специфіка деяких додатків вимагає використання команд обробки даних низького рівня. З цією метою СУБД оснащуються *процедурними* засобами обробки даних.

Зокрема, в стандарт SQL входить процедурне розширення мови, яке підтримується багатьма сучасними СУБД (правда, зі значними відхиленнями від стандарту). Інший підхід полягає в тому, що в якості вхідного використовується звичайну мову програмування, розширений конструкціями непроцедурного SQL.

Масштабованість і переносимість.

У міру зростання інформаційних потреб і розвитку парку обчислювальної техніки перед організацією, експлуатуючої деяку СУБД, неминуче постає питання про модернізацію застосовуваної технології обробки даних. З цієї точки зору перевагою володіють СУБД, характеризуються широким діапазоном масштабованості, тобто

системи, застосування яких з технічних і економічних міркувань доцільно і можливо як в умовах скромних можливостей наявного парку ВТ, так і для побудови великомасштабних систем баз даних. Слід мати на увазі, що СУБД як програмний засіб функціонує під управлінням певної операційної системи (як кажуть, - на певній платформі), і виробники СУБД розробляють свій продукт під різні платформи. Таким чином, при переході на нову платформу споживач повинен враховувати можливості перенесення експлуатованої СУБД, наявної БД і прикладних програм на цю платформу. Залежно від функціональних можливостей СУБД тут можуть виникнути такі ситуації:

- існуюча СУБД піддається перенастроюванні, а додатки і БД залишаються без змін (найкращий варіант);
- встановлюється нова версія СУБД без (явної) модернізації БД і додатків;
- встановлюється нова версія СУБД з перезавантаженням БД, але без модифікації додатків;
- встановлюється нова версія СУБД, перезавантажується БД і перепрограмовуються додатки

Архітектура розподіленої СУБД.

Сучасні СУБД забезпечують можливість одночасної роботи з БД багатьох користувачів (в рамках локальної або територіально розподіленої обчислювальної мережі). При цьому є два різних підходи до організації такої роботи. В архітектурі *файл сервер* запити користувача обробляються локально на його робочої станції засобами локально встановленої на цій станції копії СУБД, а дані, необхідні для виконання запиту, можуть перебувати на інший робочої станції мережі. При цьому кожна локальна копія СУБД виконує роль так званого сервера БД, відстежуючи можливі одночасні звернення до одного фрагменту БД з різних додатків і встановлюючи порядок блокування і розблокування доступу до цих фрагментах. Подібна технологія застосовується зазвичай для невеликих локальних мереж і при низької інтенсивності обміну даними між робочими станціями. Архітектурі *клієнт / сервер* виділяється спеціальна серверна частина СУБД, яка зазвичай разом з основною БД розгортається на потужної обчислювальної машині, і клієнтська частина, що розгортається на робочих станціях. Обробка запиту користувача практично повністю здійснюється сервером БД, а клієнтська частина відіграє роль інтерфейсу (сполучної ланки) між додатком і сервером. Така архітектура забезпечує високу надійність і ефективність системи, так як обмін даними по мережі мінімізується, а управління доступом здійснюється централізовано. У той же час СУБД, що підтримують дану архітектуру, вимагають для своєї роботи значних обчислювальних ресурсів.

1.3. Основні вимоги до СУБД.

Якість СУБД багато в чому визначаться тим, якою мірою вона дозволяє задовольнити основні вимоги, що пред'являються до функціонування СУБД. Серед таких вимог (критеріїв) можна назвати: продуктивність, ресурси, цілісність БД, незалежність даних, безпеку, простоту використання, прозорість. Дамо коротку характеристику кожного з цих критеріїв.

Продуктивність кількісно можна оцінити як час реакції системи на запит користувача. (Природно, при цьому необхідно враховувати такі фактори, як інтенсивність одночасного звернення до фрагментів БД, обсяг БД, пропускна здатність каналів обміну і ін.). За інших рівних умов продуктивність забезпечується здатністю СУБД оптимізувати процес пошуку і оновлення даних в файлах БД, знаходити раціональний план обчислення (тобто порядок виконання елементарних операцій) запиту (системи одночасно обслуговуються запитів).

Для ефективної роботи СУБД, як правило, потрібні значні обчислювальні ресурси. В цьому відношенні особливо критичними виявляються обсяг оперативної пам'яті, пропускна здатність каналів введення / виведення і каналів обміну, продуктивність процесора, обсяг зовнішньої пам'яті. Зрозуміло, що вимоги до ресурсів безпосередньо залежать від масштабності СБД і характеру розв'язуваних у ній завдань. У той же час СУБД повинні забезпечувати економне використання наявних ресурсів. Як показує практика їх експлуатації, така вимога далеко не завжди виконується. З метою

раціонального розподілу ресурсів (що може здійснюватися і «вручну» шляхом конфігурації апаратних і програмних засобів) СУБД повинна бути оснащена *засобами моніторингу та статистики.*

Цілісність БД є основною умовою безвідмовного функціонування СБД. Практично всі сучасні СУБД оснащені потужним арсеналом засобів підтримки цілісності і парирування аварійних ситуацій. Тут головна роль відводиться засобам автоматичного копіювання і відновлення БД. Крім того, вхідні мови СУБД можуть містити команди звернення до *процедурам обробки виняткових ситуацій,* що дає користувачеві додаткові можливості забезпечення цілісності.

Незалежність даних розуміється як можливість внесення змін в структуру зберігається БД (що часто буває необхідно для підвищення ефективності її функціонування) без змін у «логічної» структури даних і прикладних програмах (запитах). На практиці неможливо досягти абсолютної незалежності даних; СУБД забезпечують лише ту чи іншу ступінь незалежності (за рахунок наявності розвинених *автоматично настроюються внутрішніх інтерфейсів* «Користувач - логічна структура» і «логічна структура - структура зберігання»).

Безпека БД розуміється як її захищеність від несанкціонованого використання. Кожен користувач (Прикладна програма) в СБД має певні права доступу до тих чи інших фрагментів БД. Для того, щоб здійснити установку цих прав і проводити їх контроль в процесі функціонування СУБД повинна бути оснащена широким арсеналом *коштів* створення підтримки *системи захисту.* Слід розуміти, що підвищення вимог до рівня безпеки призводить до додаткових витрат ресурсів і зниження продуктивності.

Простота використання - неформальний критерій, виявляється у вимогах до кваліфікації обслуговуючого персоналу, зручність роботи користувачів різних категорій (прикладнихпрограмістів, адміністратора БД, операторів підготовки даних, кінцевих користувачів - споживачів вихідний інформації). Простота використання визначається такими факторами, як наявність в СУБД дружніх користувальницьких інтерфейсів, якість і методичний рівень супровідної документації, наявність спектра вхідних мов різного рівня, прозорість вихідного мови (мови повідомлень).

Вимога прозорості даних складається в наданні користувачам можливості звернення до розподілених в мережі даними без зміни технології своєї роботи. СУБД повинна по можливості «Приховувати» від користувача додаткові технологічні деталі операцій звернення до віддалених даних.

Неважко помітити, що перераховані вимоги є часто суперечливими, і для вирішення питання вибору типу і конфігурації СУБД слід враховувати ступінь важливості кожного з вимог в конкретній обстановці. До того ж наведений тут перелік є далеко не повним. Зокрема, через специфіку питань ми не в повній мірі проаналізували вимоги до адміністрування БД, розробці додатків, адміністрування додатків і ін.

Питання для самоконтролю:

1. Перерахуйте експлуатаційні вимоги, що пред'являються до СУБД?
2. Поясніть технологію взаємодії СУБД з операційної системою персонального комп'ютера?
3. Уявіть архітектуру розподіленої СУБД.
4. У чому відмінність і загальне технологій файл / сервер і клієнт / сервер?
5. Що розуміється під інформацією?
6. Дайте визначення файлу даних.
7. Що таке база даних?
8. Що означає файлова організація даних?

Лекція 2.

Основні функції систем управління базами даних

2.1. Безпосереднє управління даними у зовнішній пам'яті.

2.2. Управління буферами оперативної пам'яті.

2.3. Управління транзакціями.
2.4. Журналізація.
2.5. Підтримка мов БД.
2.1. Безпосереднє управління даними у зовнішній пам'яті
Ця функція включає забезпечення необхідних структур зовнішньої пам'яті як для зберігання даних, які безпосередньо входять в БД, так і для службових цілей, наприклад, для прискорення доступу до даних в деяких випадках (зазвичай для цього використовуються індекси). У деяких реалізаціях СУБД активно використовуються можливості існуючих файлових систем, в інших робота проводиться аж до рівня пристроїв зовнішньої пам'яті. Але підкреслимо, що в розвинених СУБД користувачі в будь-якому разі не зобов'язані знати, чи використовує СУБД файлову систему, і якщо використовує, то як організовані файли. Зокрема, СУБД підтримує власну систему іменування об'єктів БД.

2.2. Управління буферами оперативної пам'яті

СУБД зазвичай працюють з БД значного розміру; принаймні цей розмір зазвичай істотно більше доступного обсягу оперативної пам'яті. Зрозуміло, що якщо при зверненні до будь-якого елементу даних буде здійснюватися обмін із зовнішньою пам'яттю, то вся система буде працювати зі швидкістю пристрою зовнішньої пам'яті. Практично єдиним способом реального збільшення цієї швидкості є буферизація даних в оперативній пам'яті. При цьому, навіть якщо операційна система проводить загальносистемну буферизацию (як у випадку ОС UNIX), цього недостатньо для цілей СУБД, яка має в своєму розпорядженні набагато більшою інформацією про корисність буферизації тієї чи іншої частини БД. Тому в розвинених СУБД підтримується власний набір буферів оперативної пам'яті з власною дисципліною заміни буферів.
Зауважимо, що існує окремий напрямок СУБД, яке орієнтоване на постійну присутність в оперативній пам'яті всієї БД. Цей напрямок грунтується на припущенні, що в майбутньому обсяг оперативної пам'яті комп'ютерів буде настільки великий, що дозволить не турбуватися про буферизації. Поки ці роботи знаходяться в стадії досліджень.

2.3. управління транзакціями

Транзакція - це послідовність операцій над БД, розглянутих СУБД як єдине ціле. Або транзакція успішно виконується,
і СУБД фіксує (COMMIT) зміни БД, вироблені цією транзакцією, у зовнішній пам'яті, або жодне з цих змін ніяк не відбивається на стані БД. Поняття транзакції необхідне для підтримки логічної цілісності БД. Якщо згадати наш приклад інформаційної системи з файлами СПІВРОБІТНИКИ і ВІДДІЛИ, то єдиним способом не порушити цілісність БД при виконанні операції прийому на роботу нового співробітника є об'єднання елементарних операцій над файлами СПІВРОБІТНИКИ і ВІДДІЛИ в одну транзакцію. Таким чином, підтримка механізму транзакцій є обов'язковою умовою навіть одного користувача СУБД (якщо, звичайно, така система заслуговує назви СУБД). Але поняття транзакції набагато важливіше в багатокористувацьких СУБД.
Те властивість, що кожна транзакція починається при цілісному стані БД і залишає цей стан цілісним після свого завершення, робить дуже зручним використання поняття транзакції як одиниці активності користувача по відношенню до БД. при відповідному управлінні паралельно виконуються транзакціями з боку СУБД кожен з користувачів може в принципі відчувати себе єдиним користувачем СУБД (насправді, це дещо ідеалізований погляд, оскільки в деяких випадках користувачі багатокористувацьких СУБД можуть відчути присутність своїх колег).

З управлінням транзакціями в багатокористувацької СУБД пов'язані важливі поняття *сериализации транзакцій* і *серіального плану виконання суміші транзакцій.* Під Серіалізация паралельно виконуються транзакцій розуміється такий порядок планування їх роботи, при якому сумарний ефект суміші транзакцій еквівалентний

ефекту їх деякого послідовного виконання. Серіальний план виконання суміші транзакцій - це такий план, який призводить до сериализации транзакцій.

Зрозуміло, що якщо вдається домогтися дійсно серіального виконання суміші транзакцій, то для кожного користувача, з ініціативи якого утворена транзакція, присутність інших транзакцій буде непомітно (якщо не брати до уваги деякого уповільнення роботи в порівнянні з однокористувацький режимом).
Існує кілька базових алгоритмів сериализации транзакцій. У централізованих СУБД найбільш поширені алгоритми, засновані на синхронізаційних захопленнях об'єктів БД. При використанні будь-якого алгоритму сериализации можливі ситуації конфліктів між двома або більше транзакціями з доступу до об'єктів БД. В цьому випадку для підтримки сериализации необхідно виконати відкат (Ліквідувати всі зміни, вироблені в БД) однієї або більше транзакцій. Це один з випадків, коли користувач багатокористувацької СУБД може реально (і досить неприємно) відчути присутність в системі транзакцій інших користувачів.

2.4. Журналізація

Одним з основних вимог до СУБД є надійність зберігання даних у зовнішній пам'яті. Під надійністю зберігання розуміється те, що СУБД повинна бути в змозі відновити останній узгоджений стан БД після будь-якого апаратного або програмного збою. Зазвичай розглядаються два можливі види апаратних збоїв: так звані м'які збої, які можна трактувати як раптову зупинку роботи комп'ютера (наприклад, аварійне вимкнення живлення), і жорсткі збої, що характеризуються втратою інформації на носіях зовнішньої пам'яті. Прикладами програмних збоїв можуть бути: аварійне завершення роботи СУБД (через помилки в програмі або в результаті деякого апаратного збою) чи аварійне завершення користувальницької програми, в результаті чого деяка транзакція залишається.
незавершеною. Першу ситуацію можна розглядати як особливий вид м'якого апаратного збою; при виникненні останній потрібна ліквідувати наслідки тільки однієї транзакції.
Зрозуміло, що в будь-якому випадку для відновлення БД потрібно мати у своєму розпорядженні деякою додатковою інформацією. Іншими словами, підтримку надійності зберігання даних в БД вимагає надмірності зберігання даних, причому та частина даних, яка використовується для відновлення, повинна зберігатися особливо надійно. Найбільш поширеним методом підтримання такої надмірної інформації є ведення журналу змін БД.
Журнал - це особлива частина БД, недоступна користувачам СУБД і підтримувана з особливою ретельністю (іноді підтримуються дві копії журналу, наявні на різних фізичних дисках), в яку надходять записи про всі зміни основної частини БД. У різних СУБД зміни БД журналізуются на різних рівнях: іноді запис в журналі відповідає деякої логічної операції зміни БД (наприклад, операції видалення рядка з таблиці реляційної БД), іноді - мінімальної внутрішньої операції модифікації сторінки зовнішньої пам'яті; в деяких системах одночасно використовуються обидва підходи.

У всіх випадках дотримуються стратегії "упреждающей" записи в журнал (так званого протоколу Write Ahead Log - WAL). Грубо кажучи, ця стратегія полягає в тому, що запис про зміну будь-якого об'єкта БД повинна потрапити в зовнішню пам'ять журналу раніше, ніж змінений об'єкт потрапить у зовнішню пам'ять основної частини БД. Відомо, що якщо в СУБД коректно дотримується протокол WAL, то за допомогою журналу можна вирішити всі проблеми відновлення БД після будь-якого збою.
Найпростіша ситуація відновлення - індивідуальний відкат транзакції. Строго кажучи, для цього не потрібно загальносистемний журнал змін БД. Досить для кожної транзакції підтримувати локальний журнал операцій модифікації БД, виконаних в цій транзакції, і виробляти відкат транзакції шляхом виконання зворотних операцій, слідуючи від кінця локального журналу. У деяких СУБД так і роблять, але в більшості систем локальні журнали не підтримують, а індивідуальний відкат транзакції виконують по общесистемному журналу, для чого все записи від однієї транзакції пов'язують зворотним списком (від кінця до початку).

При м'якому збої у зовнішній пам'яті основної частини БД можуть перебувати

об'єкти, модифіковані транзакціями, НЕ закінчилися до моменту збою, і можуть бути відсутніми об'єкти, модифіковані транзакціями, які до моменту збою успішно завершилися (через використання буферів оперативної пам'яті, вміст яких при м'якому збої пропадає). при дотриманні протоколу WAL у зовнішній пам'яті журналу повинні гарантовано перебувати записи, які відносяться до операцій модифікації обох видів об'єктів. Метою процесу відновлення після м'якого збою є стан зовнішньої пам'яті основної частини БД, яке виникло б при фіксації в зовнішній пам'яті змін всіх завершилися транзакцій і яке не містило б ніяких слідів незакінчених транзакцій. Для того, щоб цього домогтися, спочатку виробляють відкат незавершених транзакцій (undo), а потім повторно відтворюють (redo) ті операції завершених транзакцій, результати яких не відображені у зовнішній пам'яті. Цей процес містить багато тонкощів, пов'язаних із загальною організацією управління буферами і журналом. Більш детально ми розглянемо це у відповідній лекції.

Для відновлення БД після жорсткого збою використовують журнал і архівну копію БД. Грубо кажучи, архівна копія - це повна копія БД до моменту початку заповнення журналу (мається багато варіантів більш гнучкою трактування сенсу архівної копії). Звичайно, для нормального відновлення БД після жорсткого збою необхідно, щоб журнал не пропав. Як вже зазначалося, до збереження журналу в зовнішній пам'яті в СУБД пред'являються особливо підвищені вимоги. Тоді відновлення БД полягає в тому, що, виходячи з архівної копії, за журналом відтворюється робота всіх транзакцій, які закінчилися до моменту збою. В принципі, можна навіть відтворити роботу незавершених транзакцій і продовжити їх роботу після завершення відновлення. Однак в реальних системах це зазвичай не робиться,

2.5. Підтримка мов БД

Для роботи з базами даних використовуються спеціальні мови, в цілому звані *мовами баз даних.* У ранніх СУБД підтримувалося декілька спеціалізованих за своїми функціями мов. Найчастіше виділялися дві мови - *мова визначення схеми* БД *(SDL - Schema Definition Language)* і *мова маніпулювання даними (DML - Data Manipulation Language).* SDL служив головним чином для визначення логічної структури БД, тобто тієї структури БД, якою вона представляється користувачам. DML містив набір операторів маніпулювання даними, тобто операторів, що дозволяють заносити дані в БД, видаляти, модифікувати або вибирати існуючі дані. Ми розглянемо більш докладно мови ранніх СУБД в наступній лекції.

В сучасних СУБД зазвичай підтримується єдиний інтегрований мову, що містить всі необхідні засоби для роботи з БД, починаючи від її створення, і забезпечує базовий призначений для користувача інтерфейс з базами даних. Стандартним мовою найбільш поширених в даний час реляційних СУБД є мова SQL (Structured Query Language). У кількох лекціях цього курсу мову SQL буде розглядатися досить докладно, а поки ми перерахуємо основні функції реляційної СУБД, підтримувані на "мовному" рівні (тобто функції, підтримувані при реалізації інтерфейсу SQL).

Перш за все, мова SQL поєднує засоби SDL і DML, тобто дозволяє визначати схему реляційної БД і маніпулювати даними. При цьому іменування об'єктів БД (для реляційної БД - іменування таблиць і їх стовпців) підтримується на мовному рівні в тому сенсі, що компілятор мови SQL виробляє перетворення імен об'єктів в їх внутрішні ідентифікатори на підставі спеціально підтримуваних службових таблиць-каталогів. Внутрішня частина СУБД (ядро) взагалі не працює з іменами таблиць і їх стовпців.

Мова SQL містить спеціальні засоби визначення обмежень цілісності БД. Знову ж, обмеження цілісності зберігаються в спеціальних таблицях-каталогах, і забезпечення контролю цілісності БД проводиться на мовному рівні, тобто при компіляції операторів модифікації БД компілятор SQL на підставі наявних у БД обмежень цілісності генерує відповідний програмний код.

Спеціальні оператори мови SQL дозволяють визначати так звані представлення БД, що фактично є збереженими в БД запитами (результатом будь-якого запиту до реляційної БД є таблиця) з іменованими стовпцями. Для користувача подання є такою ж таблицею,

як будь-яка базова таблиця, яка зберігається в БД, але за допомогою уявлень можна обмежити або навпаки розширити видимість БД для конкретного користувача. Підтримка уявлень проводиться також на мовному рівні.
Нарешті, авторизація доступу до об'єктів БД виробляється також на основі спеціального набору операторів SQL. Ідея полягає в тому, що для виконання операторів SQL різного виду користувач повинен володіти різними повноваженнями. Користувач, який створив таблицю БД, володіє повним набором повноважень для роботи з цією таблицею. У число цих повноважень входить повноваження на передачу всіх або частини повноважень іншим користувачам, включаючи повноваження на передачу повноважень. Повноваження користувачів описуються в спеціальних таблицях-каталогах, контроль повноважень підтримується на мовному рівні.

Питання для самоконтролю:

2. Які основні функції виконує СУБД? Перерахуйте призначення і основні функції мов SQL, SDL і DML.
3. У чому сенс журналізації?
4. Навіщо і в яких випадках необхідно управляти транзакціями?
5. Поясніть порядок взаємодії СУБД з операційною системою персонального комп'ютера?
6. Яку модель даних підтримує СУБД MS Access?
7. Які функції виконує адміністратор бази даних?
8. Які функції виконує адміністратор предметної області?

Лекція 3.

Ієрархічна і мережна моделі даних

3.1. Особливості побудови СУБД.
3.2. Ієрархічні системи.
3.3. Мережеві системи.

3.1. Особливості побудови СУБД.

Організація типовою СУБД і склад її компонентів відповідає розглянутому нами набору функцій. Нагадаємо, що ми виділили такі основні функції СУБД:

- управління даними у зовнішній пам'яті;
- управління буферами оперативної пам'яті;
- управління транзакціями;
- журнализация і відновлення БД після збоїв;
- підтримка мов БД.

Логічно в сучасній реляційної СУБД можна виділити найбільш внутрішню частину - ядро СУБД (часто його називають Data Base Engine), компілятор мови БД (зазвичай SQL), підсистему підтримки часу виконання, набір утиліт. У деяких системах ці частини виділяються явно, в інших - ні, але логічно такий поділ можна провести у всіх СУБД.
Ядро СУБД відповідає за управління даними у зовнішній пам'яті, управління буферами оперативної пам'яті, управління транзакціями і журнализацию. Відповідно, можна виділити такі компоненти ядра (принаймні, логічно, хоча в деяких системах ці компоненти виділяються явно), як менеджер даних, менеджер буферів, менеджер транзакцій і менеджер журналу. Як можна було зрозуміти з першої частини цієї лекції, функції цих компонентів між собою, і для забезпечення коректної роботи СУБД всі ці компоненти повинні взаємодіяти з ретельно продуманим і перевіреним протоколах. Ядро СУБД має власний інтерфейс, який недоступний користувачам безпосередньо використовуються в програмах, вироблених компілятором SQL (або в підсистемі підтримки виконання таких програм) і утиліти БД. Ядро СУБД є основною резидентної частиною СУБД. При використанні архітектури "клієнт-сервер" ядро є основною складовою серверної частини системи.
Основною функцією компілятора мови БД є компіляція операторів мови БД в деяку виконувану програму. Основною проблемою реляційних СУБД є те, що мови цих

систем (а це, як правило, SQL) є непроцедурного, тобто в операторі такої мови специфицируется деяку дію над БД, але ця специфікація не є процедурою, а лише описує в деякій формі умови здійснення бажаного дії (згадайте приклади з першої лекції). Тому компілятор повинен вирішити, яким чином виконувати оператор мови перш, ніж зробити програму. Застосовуються досить складні методи оптимізації операторів, які ми детально розглянемо в наступних лекціях. Результатом компіляції є виконувана програма, яка надається в деяких системах в машинних кодах, але більш часто в виконуваному внутрішньому машинно-незалежній коді.
Нарешті, в окремі утиліти БД звичайно виділяють такі процедури, які занадто накладно виконувати з використанням мови БД, наприклад, завантаження і вивантаження БД, збір статистики, глобальна перевірка цілісності БД і т.д. Утиліти програмуються з використанням інтерфейсу ядра СУБД, а іноді навіть з проникненням всередину ядра.

3.2. Ієрархічні системи.

Типовим представником (найбільш відомим і поширеним) є Information Management System (IMS) фірми IBM. Перша версія з'явилася в 1968 р До сих пір підтримується багато баз даних, що створює істотні проблеми з переходом як на нову технологію БД, так і на нову техніку.
Ієрархічна БД складається з упорядкованого набору дерев; більш точно, з упорядкованого набору кількох примірників одного типу дерева.
Тип дерева складається з одного "кореневого" типу запису і упорядкованого набору з нуля або більше типів піддерев (кожне з яких є певним типом дерева). Тип дерева в цілому являє собою ієрархічно організований набір типів записи.
Приклад типу дерева (схеми ієрархічної БД):

Рис.3.1. Фрагмент схеми ієрархічної БД Тут Відділ є предком для Начальник і Співробітники, а Начальник і Співробітники - нащадки Відділ. Між типами записи підтримуються зв'язки.
База даних з такою схемою могла б виглядати наступним чином (ми показуємо один екземпляр дерева):

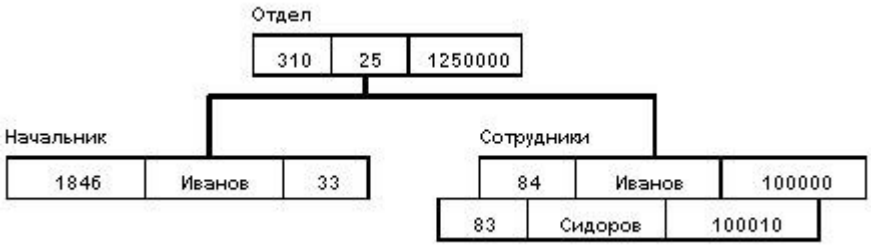


Рис.3.2. Примірник дерева ієрархічної БД Всі екземпляри даного типу нащадка із загальним примірником типу предка називаються близнюками. Для БД визначений повний порядок обходу - зверху-вниз, зліва-направо.
У IMS використовувалася оригінальна і нестандартна термінологія: "сегмент" замість "запис", а під "записом БД" розумілося все дерево сегментів. *Маніпулювання даними.*

Прикладами типових операторів маніпулювання ієрархічно організованими даними можуть бути наступні:

- Знайти вказане дерево БД (наприклад, відділ 310);

- Перейти від одного дерева до іншого;
- Перейти від одного запису до іншого всередині дерева (наприклад, від відділу - допершого співробітника);
- Перейти від одного запису до іншого в порядку обходу ієрархії;
- Вставити новий запис у вказану позицію;
- Видалити поточний запис.

Обмеження цілісності.

Автоматично підтримується цілісність посилань між предками і нащадками. *Основне правило: ніякої нащадок не може існувати без свого батька.* Зауважимо, що аналогічне підтримку цілісності по посиланнях між записами, що не входять в одну ієрархію, не підтримуються (прикладом такої "зовнішньої" посилання може бути вміст поля Каф_Номер в екземплярі типу записи Куратор).

В ієрархічних системах підтримувалася деяка форма уявлень БД на основі обмеження ієрархії. Прикладом уявлення наведеної вище БД може бути ієрархія

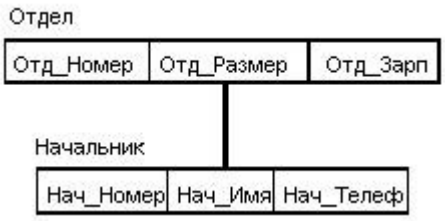


Рис.3.3. Приклад зв'язку в ієрархічній БД

3.3. Мережеві системи.

Типовим представником є Integrated Database Management System (IDMS) компанії Cullinet Software, Inc., призначена для використання на машинах основного класу фірми IBM під управлінням більшості операційних систем. Архітектура системи заснована на пропозиціях Data Base Task Group (DBTG) Комітету з мов програмування Conference on Data Systems Languages (CODASYL), організації, відповідальної за визначення мови програмування Кобол. Звіт DBTG був опублікований в 1971 р, а в 70-х роках з'явилося кілька систем, серед яких IDMS.

Мережеві структури даних.

Мережевий підхід до організації даних є розширенням ієрархічного. В ієрархічних структурах запис-нащадок повинна мати в точності одного предка; в мережевій структурі даних нащадок може мати будь-яке число предків.

Мережева БД складається з набору записів і набору зв'язків між цими записами, а якщо говорити більш точно, з набору примірників кожного типу із заданого в схемі БД набору типів запису і набору примірників кожного типу із заданого набору типів зв'язку. Тип зв'язку визначається для двох типів запису: предка і нащадка. Примірник типу зв'язку складається з одного примірника типу записи предка і упорядкованого набору примірників типу записи нащадка. Для даного типу зв'язку L з типом запису предка P і типом запису нащадка C повинні виконуватися наступні дві умови:

- Кожен екземпляр типу P є предком тільки в одному екземплярі L;
- Кожен екземпляр C є нащадком не більше, ніж в одному екземплярі L.

На формування типів зв'язку не накладаються особливі обмеження; можливі, наприклад, такі ситуації:

- Тип запису нащадка в одному типі зв'язку L1 може бути типом запису предка в іншому типі зв'язку L2 (як в ієрархії).
- Даний тип запису P може бути типом запису предка в будь-якій кількості типів зв'язку.
- Даний тип запису P може бути типом запису нащадка в будь-якій кількості типів зв'язку.

- Може існувати будь-яке число типів зв'язку з одним і тим же типом запису предка і одним і тим же типом запису нащадка; і якщо L1 і L2 - два типи зв'язку

з одним і тим же типом запису предка P і одним і тим же типом запису нащадка C, то правила, за якими утворюється співвідношення, в різних зв'язках можуть відрізнятися.

- Типи записи X і Y можуть бути предком і нащадком в одній зв'язку і нащадком і предком - в інший.
- Предок і нащадок можуть бути одного типу запису. Простий приклад мережевий схеми БД:

Рис.3.4. Фрагмент схеми ієрархічної БД *Маніпулювання даними.*

Зразковий набір операцій може бути наступним:

- Знайти певний запис в наборі однотипних записів (інженера Сидорова);
- Перейти від предка до першого нащадку за деякою зв'язку (до першого співробітнику відділу 310);
- Перейти до наступного нащадку у деякому зв'язку (від Сидорова до Іванова);
- Перейти від нащадка до предка за деякою зв'язку (знайти відділ Сидорова);
- Створити новий запис;
- Знищити запис;
- Змінити запис;
- Включити в зв'язок;
- Виключити з зв'язку;
- Переставити в інший зв'язок і т.д.

Обмеження цілісності.

В принципі їх підтримку не потрібно, але іноді вимагають цілісності по посиланнях (як в ієрархічній моделі).

Питання для самоконтролю:

1. Які Ви знаєте типи моделей даних?
2. Що розуміється під інформацією?
3. Дайте визначення ієрархічної моделі даних
4. Що розуміється під структурою даних?
5. Які Ви знаєте моделі даних інформаційних систем?
6. Визначення слабо типизированной моделі
7. Дайте визначення мережевої моделі даних

Лекція 4.

Реляційна модель даних

4.1. Основи реляційних баз даних.
4.2. Базові поняття реляційних баз даних
4.3. Типи даних.

4.4. Поняття домен в реляційних системах.
4.5. Схема відносини, схема бази даних.

4.6. Кортеж, визначення реляційного відносини

4.1. Основи реляційних бази даних.

Ми приступаємо до вивчення реляційних баз даних і систем керування базами даних. Цей підхід є найбільш поширеним в даний час, хоча поряд з загальновизнаними достоїнствами володіє і рядом недоліків. До достоїнств реляційного підходу можна віднести:

- наявність невеликого набору абстракцій, які дозволяють порівняно просто моделювати більшу частину поширених предметних областей і допускають точні формальні визначення, залишаючись інтуїтивно зрозумілими;

- наявність простого і в той же час потужного математичного апарату, що спирається головним чином на теорію множин і математичну логіку і забезпечує теоретичний базис реляційного підходу до організації баз даних;
- можливість ненавігаційній маніпулювання даними без необхідності знання конкретної фізичної організації баз даних у зовнішній пам'яті.

Реляційні системи далеко не відразу отримали широке поширення. У той час, як основні теоретичні результати в цій області були отримані ще в 70-х, і тоді ж з'явилися перші прототипи реляційних СУБД, довгий час вважалося неможливим домогтися ефективної реалізації таких систем. Однак зазначені вище переваги і поступове накопичення методів і алгоритмів організації реляційних баз даних і управління ними привели до того, що вже в середині 80-х років реляційні системи практично витіснили зі світового ринку ранні СУБД.

В даний час основним предметом критики реляційних СУБД є не їх недостатня ефективність, а притаманна цим системам деяка обмеженість (прямий наслідок простоти) при використання в так званих нетрадиційних областях (найбільш поширеними прикладами є системи автоматизації проектування). Ще одним часто відзначається недоліком реляційних баз даних є неможливість адекватного відображення семантики предметної області. Іншими словами, можливості подання знань про семантичної специфіці предметної області в реляційних системах дуже обмежені. Сучасні дослідження в області постреляціонних систем головним чином присвячені саме усунення цих недоліків.

4.2. Базові поняття реляційних баз даних

Основними поняттями реляційних баз даних є тип даних, домен, атрибут, кортеж, первинний ключ і ставлення.

Для початку покажемо зміст цих понять на прикладі відношення СПІВРОБІТНИКИ, що містить інформацію про співробітників деякої організації:

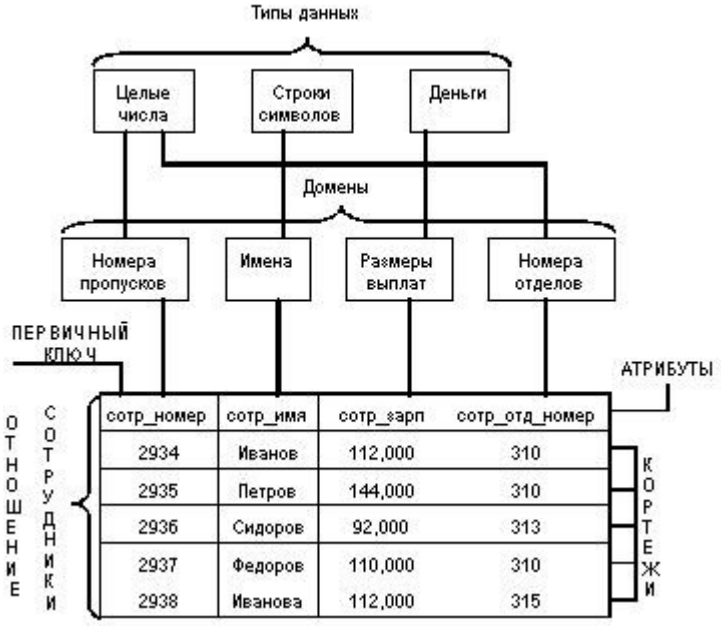


Рис.4.1. Основні елементи реляційної БД

4.3. Тип даних

поняття *тип даних* в реляційної моделі даних повністю адекватно поняттю типу даних в мовах програмування. Зазвичай в сучасних реляційних БД допускається зберігання символьних, числових даних, бітових рядків, спеціалізованих числових даних (таких як "гроші"), а також спеціальних "темпоральних" даних (дата, час, часовий інтервал). Досить активно розвивається підхід до розширення можливостей реляційних систем абстрактними типами даних (відповідними можливостями володіють, наприклад, системи сімейства Ingres / Postgres). У нашому прикладі ми маємо справу з даними трьох типів: рядки символів, цілі числа і "гроші".

4.4. Поняття домен в реляційних системах поняття *домену* більш специфічно для баз даних, хоча і має деякі аналогії з підтипами в деяких мовах програмування. У найзагальнішому вигляді домен визначається завданням деякого базового типу даних, до якого відносяться елементи домену, і довільного логічного виразу, який застосовується до елементу типу даних. Якщо обчислення цього логічного виразу дає результат "істина", то елемент даних є елементом домену.

Найбільш правильною інтуїтивною трактуванням поняття домену є розуміння домену як допустимого потенційного безлічі значень даного типу. Наприклад, домен "Імена" в нашому прикладі визначено на базовому типі рядків символів, але число його значень можуть входити тільки ті рядки, які можуть зображати ім'я (зокрема, такі рядки не можуть починатися з м'якого знака).

Слід зазначити також семантичне навантаження поняття домену: дані вважаються порівнянними тільки в тому випадку, коли вони відносяться до одного домену. У нашому прикладі значення доменів "Номери перепусток" і "Номери груп" належать до типу цілих чисел, але не є порівнянними.

4.5. Схема відносини, схема бази даних

Схема відносини - це іменоване безліч пар {ім'я атрибута, ім'я домену (або типу, якщо поняття домену не підтримується)}. Ступінь або "арность" схеми відносини - потужність цієї множини. Ступінь відносини СПІВРОБІТНИКИ дорівнює чотирьом, тобто воно є 4-арним. Якщо всі атрибути одного відносини визначені на різних доменах, осмислено використовувати для іменування атрибутів імена відповідних доменів (не забуваючи, звичайно, про те, що це є всього лише зручним способом іменування і не усуває відмінності між поняттями домену і атрибута).

Схема БД (в структурному сенсі) - це набір іменованих схем відносин.

4.6. Кортеж, визначення реляційного відносини

Кортеж, що відповідає даній схемі відносини- це безліч пар {ім'я атрибута, значення}, яке містить одне входження кожного імені атрибута, що належить схемою відносини. "Значення" є допустимим значенням домену даного атрибута (або типу даних, якщо поняття домену не підтримується). Тим самим, ступінь або "арность" кортежу, тобто число елементів в ньому, збігається з "арность" відповідної схеми відносини. Попросту кажучи, кортеж - це набір іменованих значень заданого типу.

Ставлення - це безліч кортежів, що відповідають одній схемі відносини. Іноді, щоб не плутатися, говорять "отношеніе- схема" і "ставлення-екземпляр", іноді схему відносини називають заголовком відносини, а відношення як набір кортежів - тілом відносини. Насправді, поняття схеми відносини найближче до поняття структурного типу даних в мовах програмування. Було цілком логічно дозволяти окремо визначати схему відносини, а потім одне або кілька відносин з даною схемою.

Однак в реляційних базах даних це не прийнято. Ім'я схеми відносини в таких базах даних завжди збігається з ім'ям відповідного ставлення-екземпляра. У класичних реляційних базах даних після визначення схеми бази даних змінюються тільки відносини-екземпляри. У них можуть з'являтися нові і віддалятися або модифікуватися існуючі кортежі. Однак у багатьох реалізаціях допускається і зміна схеми бази даних: визначення нових та зміна існуючих схем відносини. Це прийнято називати *еволюцією схеми бази даних.*

Звичайним життєвим поданням відносини є таблиця, заголовком якої є схема відносини, а рядками - кортежі відносини-екземпляри; в цьому випадку імена атрибутів іменують стовпці цієї таблиці. Тому іноді говорять "стовпець таблиці", маючи на увазі "атрибут відносини". Коли ми перейдемо до розгляду практичних питань організації реляційних баз даних і засобів управління, ми будемо використовувати цю життєву термінологію. Цією термінології дотримуються в більшості комерційних реляційних СУБД.
Реляційна база даних - це набір відносин, імена яких збігаються з іменами схем відносин в схемі БД.
Як видно, основні структурні поняття реляційної моделі даних (якщо не брати до уваги поняття домену) мають дуже просту інтуїтивну інтерпретацію, хоча в теорії реляційних БД всі вони визначаються абсолютно формально і точно.

Питання для самоконтролю:

1. На якій структурі заснована реляційна модель даних?
2. Дайте визначення поняттю атрибут?
3. Дайте визначення поняттю домен?
4. Дайте визначення поняттю кортеж?
5. Дайте визначення поняттю сегмент?
6. Що таке потужність відносини?

Розділ 2. Принципи проектування баз даних

Лекція 5.

Інфологічна модель даних "сутність-зв'язок"

5.1. Основні поняття.
5.2. Характеристика зв'язків і мова моделювання
5.3. Класифікація сутностей
5.1. Основні поняття

Мета інфологічне моделювання - забезпечення найбільш природних для людини способів збору і представлення тієї інформації, яку передбачається зберігати в створюваній базі даних. Тому інфологічну модель даних намагаються будувати за аналогією з природною мовою (останній не може бути використаний в чистому вигляді через складність комп'ютерної обробки текстів і неоднозначності будь-якого природного мови). основними конструктивними елементами інфологічних моделей є сутності, зв'язки між ними і їх властивості (атрибути). *сутність* - будь-який помітний об'єкт (об'єкт, який ми можемо відрізнити від іншого), інформацію про який необхідно зберігати в базі даних. Сутностями можуть бути люди, місця, літаки, рейси, смак, колір і т.д. Необхідно розрізняти такі поняття, як *тип сутності* і *екземпляр сутності.* Поняття тип сутності відноситься до набору однорідних особистостей, предметів, подій або ідей, виступаючих як ціле. Примірник сутності відноситься до конкретної речі в наборі. Наприклад, типом сутності може бути МІСТО, а екземпляром - Москва, Київ і т.д.
Атрибут - пойменована характеристика сутності. Його найменування повинно бути унікальним для конкретного типу сутності, але може бути однаковим для різного типу сутностей (наприклад, КОЛІР може бути визначений для багатьох сутностей: СОБАКА, АВТОМОБІЛЬ, ДИМ і т.д.). Атрибути використовуються для визначення того, яка інформація повинна бути зібрана про сутність. Прикладами атрибутів для сутності АВТОМОБІЛЬ є ТИП, МАРКА, НОМЕРНИЙ знак, колір і т.д. Тут також існує відмінність між типом і екземпляром.
Тип атрибута КОЛІР має багато екземплярів чи значень: Червоний, Синій, Банановий, Біла ніч і т.д., однак кожному екземпляру сутності привласнюється тільки одне значення атрибута.

Абсолютна відмінність між типами сутностей і атрибутами відсутня. Атрибут є таким тільки в зв'язку з типом

суті. В іншому контексті атрибут може виступати як самостійна сутність. Наприклад, для автомобільного заводу колір - це тільки атрибут продукту виробництва, а для лакофарбової фабрики колір - тип сутності.

Ключ - мінімальний набір атрибутів, за значеннями яких можна, можливооднозначно знайти необхідний екземпляр суті. Мінімальність означає, що виключення з набору будь-якого атрибута не дозволяє ідентифікувати сутність по що залишилися. Для сутності Розклад ключем є атрибут НОМЕР_РЕЙСА або набір: Пункт_отправленія, ВРЕМЯ_ВИЛЕТА і Пункт_назначенія (за умови, що з пункту в пункт вилітає в кожен момент часу один літак).

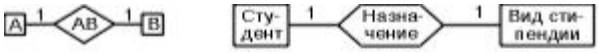
зв'язок - асоціювання двох або більше сутностей. Якби призначенням бази даних було тільки збереження окремих, не пов'язаних між собою даних, то її структура могла б бути дуже простий. Проте одна з основних вимог до організації бази даних - це забезпечення можливості відшукання одних сутностей за значеннями інших, для чого необхідно встановити між ними певні зв'язки. А так як в реальних базах даних нерідко містяться сотні або навіть тисячі сутностей, то теоретично між ними може бути встановлено більше мільйона зв'язків. Наявність такої великої кількості зв'язків і визначає складність інфологічних моделей.

5.2. Характеристика зв'язків і мова моделювання

При побудові інфологічних моделей можна використовувати мову *ER-діаграм в*ід англ. Entity-Relationship, тобто сутність-зв'язок). У них сутності зображуються позначеними прямокутниками, асоціації - позначеними ромбами або шестикутниками, атрибути - поміченими овалами, а зв'язки між ними - ненаправленої ребрами, над якими може проставлятися ступінь зв'язку (1 або буква, що заміняє слово "багато") і необхідне пояснення.

Між двома сутностей, наприклад, А і В можливі чотири види зв'язків.

перший тип - зв'язок ОДИН-К-ОДНОМУ (1: 1): в кожен момент часу кожному представнику (екземпляру) суті А відповідає 1 або 0 представників сутності В:



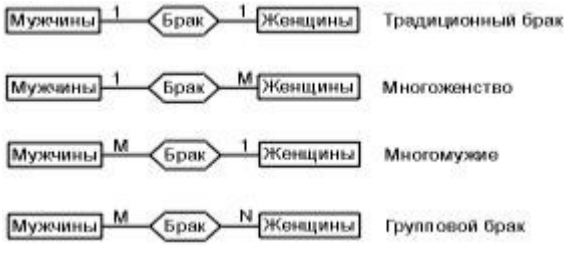
Студент може не "заробити" стипендію, одержати звичайну чи одну з підвищених стипендій.

другий тип - зв'язок ОДИН-КО-багато (1: М): одному представнику сутності А відповідають 0, 1 або кілька представників сутності В.

Квартира може пустувати, в ній може жити один чи кілька мешканців.

Так як між двома сутностями можливі зв'язки в обох напрямках, то існує ще два типи зв'язку БАГАТО-К-ОДНОМУ (М: 1) і БАГАТО-КО-багато (М: N).

Приклад 5.1. Якщо зв'язок між сутностями ЧОЛОВІКА і ЖІНКИ називається ШЛЮБ, то існує чотири возможнихпредставленія такого зв'язку:



Характер зв'язків між сутностями не обмежується перерахованими. Існують і більш складні зв'язку:

- безліч зв'язків між одними і тими ж сутностями:

(Пацієнт, маючи одного лікуючого лікаря, може мати також кілька лікарів-консультантів; лікар може бути лікуючим лікарем декількох пацієнтів і може одночасно консультувати кілька інших пацієнтів);

- тренарние зв'язку:

(Лікар може призначити кілька пацієнтів на кілька аналізів, аналіз може бути призначений декількома лікарями декільком пацієнтам і пацієнт може бути призначений на кілька аналізів декількома лікарями);

У зв'язку більш високих порядків, семантика (сенс) яких іноді дуже складна. У наведених прикладах для підвищення ілюстративності розглянутих зв'язків не показані атрибути сутностей і асоціацій у всіх ER-діаграмах. Так, введення лише кількох основних атрибутів в опис шлюбних зв'язків значно ускладнить ER-діаграму.

У зв'язку з цим мова ER-діаграм використовується для побудові невеликих моделей і ілюстрації окремих фрагментів великих. Найчастіше ж застосовується менш наочний, але більш змістовний *мова інфологічне моделювання (* ЯІМ), в якому сутності й асоціації представляються пропозиціями виду:

СУТНІСТЬ (атрибут 1, атрибут 2, ..., атрибут n) АСОЦІАЦІЯ [СУТНІСТЬ S1, СУТНІСТЬ S2, ...]

(Атрибут 1, атрибут 2, ..., атрибут n)

де S - ступінь зв'язку, а атрибути, що входять в ключ, повинні бути відзначені за допомогою підкреслення.

Так, розглянутий вище приклад безлічі зв'язків між сутностями, може бути описаний на ЯІМ наступним чином:

Лікар (Номер_врача, Прізвище, Ім'я, По батькові, Спеціальність) Пацієнт (Регістраційний_номер, Номер ліжка, Прізвище,

Ім'я, По батькові, Адреса, Дата народження, Пол)

Лечащій_врач [Лікар 1, Пацієнт M]

(Номер_врача, Регістраційний_номер)

Консультант [Лікар M, Пацієнт N]

(Номер_врача, Регістраційний_номер).

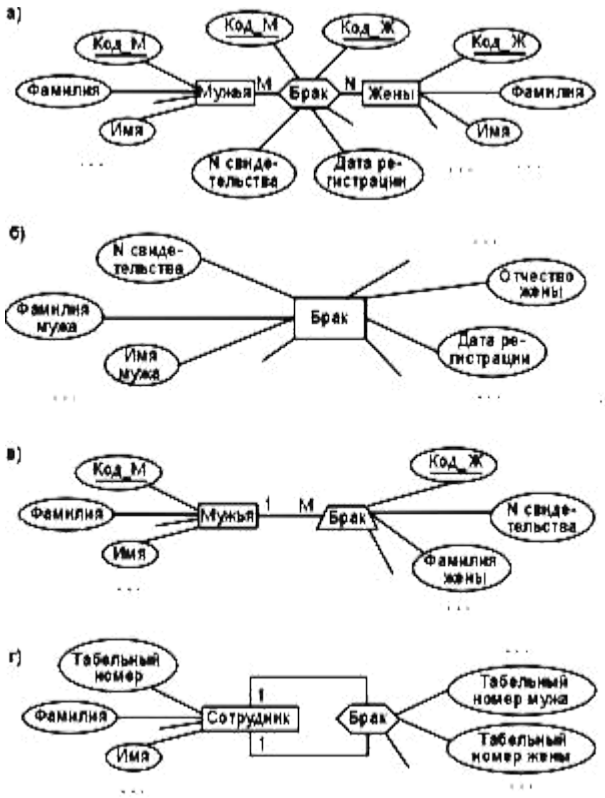


Рис.5.1. Приклади ER-діаграм

Для виявлення зв'язків між сутностями необхідно, як мінімум, визначити самі сутності. Але це не просте завдання, так як в різних предметних областях один і той же об'єкт може бути сутністю, атрибутом або асоціацією. Проілюструємо таке твердження на прикладах, пов'язаних з описом шлюбних зв'язків (див. Приклад 2.1).

Приклад 5.2. Відділ записів актів громадянського стану (РАГС) має справу не з усіма людьми, а тільки з тими, хто звернувся з проханням про реєстрацію шлюбу, народження або смерті. Тому в країнах, де допускаються лише традиційні шлюби, відділи ЗАГС можуть розміщувати відомості про реєстрованих шлюби в єдиній сутності:

Шлюб (Номер_свідетельства, Фамілія_мужа, Імя_мужа, Отчество_мужа, Дата_рожденія_мужа, Фамілія_жени, ... , Дата_регістрації, Место_регістрації, ...),

Приклад 5.3. Тепер розглянемо ситуацію, коли відділ РАГС розташований в країні, допускає багатоженство. якщо для реєстрації шлюбів використовувати сутність "Шлюб", то будуть дублюватися відомості про чоловіків, що мають кілька дружин (див. табл. 6.1).

Таблиця 5.1

Номер свидетельства	Фамилия мужа	Фамилия жены	Дата регистрации
1-ЮБ 154745	Петухов	Курочкина	06/03/1991
1-ЮБ 163489	Петухов	Пеструшкина	11/08/1991
1-ЮБ 169887	Петухов	Рябова	12/12/1992
1-ЮБ 169878	Селезнев	Уточкина	12/12/1992
1-ЮБ 154746	Парасюк	Свинюшкина	06/03/1991
1-ЮБ 169879	Парасюк	Хавронія	12/12/1992
...	...	...	...

ї

6.3. Класифікація сутностей

Настав момент розібратися в термінології. К.Дейт визначає три основні класу сутностей: *стрижневі, асоціативні* і *характеристичні,* а також підклас асоціативних сутностей -*позначення. стрижнева сутність (Стрижень)* - це незалежна сутність (Трохи докладніше вона буде визначена нижче).

У розглянутих раніше прикладах стрижні - це "Студент", "Квартира", "Чоловіки", "Лікар", "Шлюб" та інші, назви яких поміщені в прямокутники.

Асоціативна сутність (Асоціація) - це зв'язок виду "багато-ко-многим "(" ко-многим "і т.д.) між двома або більше сутностями або екземплярами сутності. Асоціації розглядаються як повноправні суті: вони можуть брати участь в інших асоціаціях і позначеннях точно так же, як стрижневі сутності; можуть мати властивості, тобто мати не тільки набір ключових атрибутів, необхідних для вказівки зв'язків, а й будь-яке число інших атрибутів, що характеризують зв'язок. Наприклад, асоціації "Шлюб" містять ключові атрибути "Код_М", "Код_Ж" і "Табельний номер чоловіка", "Табельний номер дружини", а також уточнюючі атрибути "Номер свідоцтва "," Дата реєстрації "," Место_регістрації "," Номер запису в книгу ЗАГС "і т.д. *Характеристична сутність (Характеристика)* - це зв'язок виду "багато-до-одного" або "одна-до-одного" між двома сутностями (окремий випадок асоціації). Єдина мета характеристики в рамках розглянутої предметної області полягає в описі чи уточнення деякої іншої сутності. Необхідність в них виникає в зв'язку з тим, що сутності реального світу мають іноді багатозначні властивості. Чоловік може мати кілька дружин, книга - кілька характеристик перевидання (Виправлене, доповнене, перероблене, ...) і т.д.

існування Характеристики повністю залежить від характеризується сутності: жінки позбавляються статусу жінок, якщо вмирає їх чоловік.

Для опису характеристики використовується нова пропозиція ЯІМ, що має в загальному випадку вид:

ХАРАКТЕРИСТИКА (атрибут 1, атрибут 2, ...)

{СПИСОК сутностей, що характеризується}. Розширимо також мову ER-діаграм, ввівши для зображення характеристики трапецію.

Рис. 5.2. Елементи розширеного мови ER-діаграм *означає сутність* або *позначення* - це зв'язок виду "Багато-до-одного" або "одна-до-одного" між двома сутностями і відрізняється від характеристики тим, що не залежить від позначається сутності.

Розглянемо приклад, пов'язаний із зарахуванням співробітників в різні відділи організації. При відсутності жорстких правил (співробітник може одночасно зараховуватися кілька відділів або НЕ зараховуватися ні в один відділ) необхідно створити опис з асоціацією Зарахування: Відділи (Номер відділу, Назва відділу, ...) Службовці (Табельний номер, Прізвище, ...) Зарахування [відділи М, Службовці N] (Номер відділу, Табельний номер, Дата зарахування).

Однак, за умови, що кожен із співробітників повинен бути обов'язково зарахований в один з відділів, можна створити опис з позначенням Службовці:

Відділи (Номер відділу, Назва відділу, ...) Службовці (Табельний номер, Прізвище, ..., Номер відділу, Дата зарахування) [Відділи]

В даному прикладі службовці мають незалежне існування (якщо видаляється відділ, то з цього не випливає, що також повинні бути видалені службовці такого відділу). Тому вони не можуть бути характеристиками відділів і названі позначеннями.

Позначення використовують для зберігання значень, що повторюються великих текстових атрибутів: "кодифікатори" вивчаються студентами дисциплін, найменувань організацій і їх відділів, переліків товарів і т.п. Опис позначення зовні відрізняється від опису характеристики тільки тим, що позначаються суті полягає не фігурні дужки, а в квадратні:
Ідентифікація (атрибут 1, атрибут 2, ...) [СПИСОК Позначається сутності]. Як правило, позначення не розглядаються як повноправні суті, хоча це не привело б до будь-якої помилку.
Позначення і характеристики не є повністю незалежними сутностями, оскільки вони припускають наявність деякої іншої сутності, яка буде "позначатися" або "характеризуватися". Однак вони все ж є окремі випадки сутності і можуть, звичайно, мати властивості, можуть брати участь в асоціаціях, позначеннях і мати свої власні (нижчого рівня) характеристики. Підкреслимо також, що всі екземпляри характеристики повинні бути обов'язково пов'язані з будь-яким екземпляром характеризується сутності. Однак допускається, щоб деякі екземпляри характеризується суті не мали зв'язків. Правда, якщо це стосується шлюбів, то сутність "Чоловіки" повинна бути замінена на сутність "Чоловіки" (немає чоловіка без дружини).
Перевизначити тепер стрижневу сутність як сутність, яка не є ні асоціацією, ні позначенням, ні характеристикою. Такі суті мають незалежне існування, хоча вони і можуть позначати інші сутності, як, наприклад, співробітники позначають відділи.
На закінчення розглянемо приклад побудови інфологічної моделі бази даних "Харчування", де повинна зберігатися інформація про блюдах (рис. 6.3), їх щоденному споживанні, продуктах, з яких готуються ці страви, і постачальників цих продуктів. Інформація буде використовуватися кухарем і керівником невеликого підприємства громадського харчування, а також його відвідувачами.
За допомогою зазначених користувачів виділені наступні об'єкти і характеристики проектованої бази:

1. Страви, для опису яких потрібні дані, що входять до їх кулінарні рецепти: номер страви (наприклад, з книги кулінарних рецептів), назва страви, вид блюда (закуска, суп, гаряче і т.п.), рецепт (технологія приготування страви), вихід (вага порції), назва, калорійність і вага кожного продукту, що входить в блюдо.
2. Для кожного постачальника продуктів: найменування, адреса, назва поставляється продукту, дата поставки і ціна на момент поставки.
3. Щоденне споживання страв (витрата): страва, кількість порцій, дата.

Аналіз об'єктів дозволяє виділити:

- стрижні Страви, Продукти та Міста;
- асоціації Склад (пов'язує Страви з Продуктами) і Постачання (пов'язує Постачальників з Продуктами);
- позначення Постачальники;
- характеристики Рецепти і Витрата.

ER-діаграма моделі показана на рис.6.4. а модель на мові ЯІМ має наступний вигляд:
Страви (БЛ, Блюдо, Вид)
Продукти (ПР, Продукт, Калорійність) Постачальники (ПОС, Місто, Постачальник) [Місто] Склад [Страви M, Продукти N] (БЛ, ПР, Вага (г))
Поставки [Постачальники M, Продукти N] (ПОС, ПР, Дата_П, Ціна, Вага (кг)) Міста (Місто, Країна) Рецепти (БЛ, Рецепт)
{Страви} Витрата (БЛ, Дата_Р, Працюй)
{Страви}

У цих моделях Страва, Продукт і Постачальник - найменування, а БЛ, ПР і ПОС - цифрові коди страв, продуктів і організацій, що постачають ці продукти.

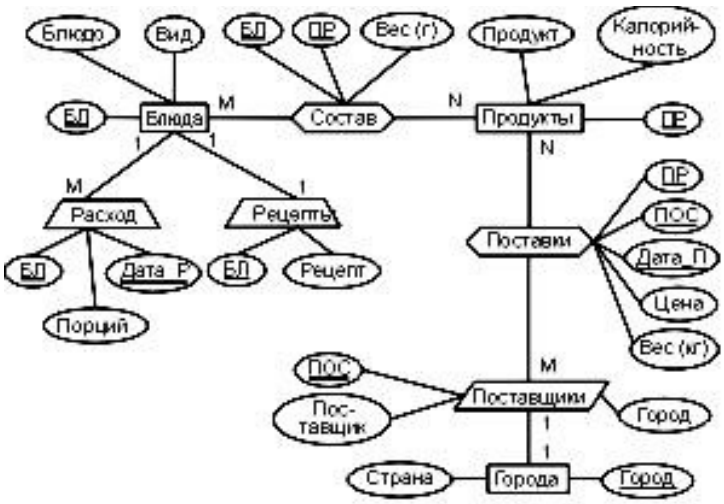


Рис.5.4. Инфологическая модель бази даних "Харчування"

Питання для самоконтролю:

1. Які бувають типи зв'язків між даними щодо?
2. Що таке унарна зв'язок?
3. Що таке бінарна зв'язок?
4. Що таке тернарного зв'язок?
5. Визначення моделі «Сутність-зв'язок»

6. Наведіть основні образотворчі кошти моделі «Сутність-зв'язок».

7. Уявіть модель «Сутність-зв'язок» для предметної області "БІБЛІОТЕКА".

Лекція 6.

Первинні і зовнішні ключі етапи розробки бази даних

6.1. Обмеження цілісності.
6.2. Проектування схеми бази даних
6.3. Етапи проектування бази даних
6.4. Критерії оцінки якості логічної моделі даних.

Нагадаємо, що *ключ* або *можливий ключ* - це мінімальний набір атрибутів, за значеннями яких можна однозначно знайти необхідний екземпляр сутності. Мінімальність означає, що виключення з набору будь-якого атрибута не дозволяє ідентифікувати сутність по що залишилися. Кожна сутність має хоча б одним можливим ключем. Один з них приймається за *первинний ключ.* При виборі первинного ключа слід віддавати перевагу несоставнимі ключам або ключам, складеним з мінімального числа атрибутів. Недоцільно також використовувати ключі з довгими текстовими значеннями (краще використовувати цілочисельні атрибути).

Так, для ідентифікації студента можна використовувати або унікальний номер залікової книжки, або набір із прізвища, імені, по батькові, номера групи і може бути додаткових атрибутів, так як не виключена поява в групі двох студентів (а частіше студенток) з однаковими прізвищами, іменами та батькові. Погано також використовувати в якості ключа не номер страви, а його назва, наприклад, "Закуска з плавлених сирків" Дружба "з шинкою і солоним огірком" або "Заєць в сметані з картопляними крокетами і салатом з червоної капусти".

Не допускається, щоб первинний ключ стрижневий суті (будь-який атрибут, який бере участь в первинному ключі) брав невизначений значення. Інакше виникне суперечлива

ситуація: з'явиться не володіє індивідуальністю, і, отже, не існуючий екземпляр стрижневий суті. З тих же причин необхідно забезпечити унікальність *первинного ключа.*
про *зовнішні ключі:*

- Якщо сутність З пов'язує суті А і В, то вона повинна включати зовнішні ключі, відповідні первинним ключам сутностей А і В.
- Якщо сутність У позначає сутність А, то вона повинна включати зовнішній ключ, відповідний первинному ключу суті А.

Зв'язок між первинними і зовнішніми ключами сутностей ілюструється рис. 5.

Рис. 6.1. структури:

асоціації; б - позначення (характеристики) Тут для позначення будь-якої з асоційованих сутностей (стрижнів, характеристик, позначень або навіть асоціацій) використовується новий узагальнюючий термін "Мета" або "Цільова сутність".

6.1. обмеження цілісності

Цілісність (від англ. integrity -незайманість, недоторканність, безпеку, цілісність) - розуміється як правильність даних в будь-який момент часу. Але ця мета може бути досягнута лише в певних межах: СУБД не може контролювати правильність кожного окремого значення, що вводиться в базу даних (хоча кожне значення можна перевірити на правдоподібність). Наприклад, не можна виявити, що вводиться значення 5 (що представляє номер дня тижня) в дійсності має дорівнювати 3. З іншого боку, значення 9 явно буде помилковим і СУБД повинна його відкинути. Однак для цього їй слід повідомити, що номери днів тижня повинні належати набору (1,2,3,4,5,6,7).

Підтримка цілісності бази даних може розглядатися як захист даних від невірних змін або руйнувань (не плутати з незаконними змінами та руйнуваннями, які є проблемою безпеки). Сучасні СУБД мають ряд засобів для забезпечення підтримки цілісності (так само, як і засобів забезпечення підтримки безпеки).

Виділяють три групи правил цілісності:

1. Цілісність по сутностей.
3. Цілісність по посиланнях.
4. Цілісність, що визначається користувачем.

Розглянемо мотивування двох правил цілісності, загальних для будь-яких реляційних баз даних.

1. Не допускається, щоб будь-якої атрибут, який бере участь в первинному ключі, отримував невизначене значення.
2. Значення зовнішнього ключа повинно або:
1. бути рівним значенню первинного ключа мети;

2. бути повністю невизначеним, тобто кожне значення атрибута, бере участь у зовнішньому ключі має бути невизначеним.

3. Для будь-якої конкретної бази даних існує ряд додаткових специфічних правил, які відносяться до неї однієї і визначаються розробником. найчастіше всього контролюється унікальність тих чи інших атрибутів, діапазон значень (екзаменаційна оцінка від 2 до 5), приналежність набору значень (пол "М" або "Ж").

6.2. Проектування схеми баз даних

Тільки невеликі організації можуть усуспільнити дані в одній повністю інтегрованій базі даних. Найчастіше адміністратор баз даних (навіть якщо це група осіб) практично не в змозі охопити і осмислити всі інформаційні вимоги співробітників організації (тобто майбутніх користувачів системи). Тому інформаційні системи великих організацій містять кілька десятків БД, нерідко розподілених між кількома взаємопов'язаними ЕОМ різних підрозділів. (Так у великих містах створюється не одна, а кілька овочевих баз, розташованих в різних районах.)

Окремі БД можуть об'єднувати всі дані, необхідні для вирішення однієї або декількох прикладних задач, або дані, які стосуються будь-якої предметної області (наприклад, фінансів, студентам, викладачам, кулінарії і т.п.). Перші зазвичай називають *прикладними БД,* а другі - *предметними БД (*співвідносяться з предметами організації, а не з її інформаційними додатками). (Перші можна порівняти з базами матеріально-технічного постачання або відпочинку, а другі з овочевими та взуттєвими базами.)

Предметні БД дозволяють забезпечити підтримку будь-яких поточних і майбутніх додатків, оскільки набір їх елементів даних включає в себе набори елементів даних прикладних БД. Внаслідок цього предметні БД створюють основу для обробки неформалізованих, змінюються і невідомих запитів та програм (додатків, для яких неможливо заздалегідь визначити вимоги до даних). Така гнучкість і пристосованість дозволяє створювати на основі предметних БД досить стабільні інформаційні системи, тобто системи, в яких більшість змін можна здійснити без вимушеного переписування старих додатків.

Засновуючи ж проектування БД на поточних і передбачуваних додатках, можна, можливо істотно прискорити створення високоефективної інформаційної системи, тобто системи, структура якої враховує найбільш часто зустрічаються шляху доступу до даних. Тому прикладне проектування до сих пір привертає деяких розробників. Однак у міру зростання числа додатків таких інформаційних систем швидко збільшується число прикладних БД, різко зростає рівень дублювання даних і підвищується вартість їх ведення.

Таким чином, кожен з розглянутих підходів до проектування впливає на результати проектування в різних напрямках. Бажання досягти і гнучкості, і ефективності призвело до формування методології проектування, що використовує як предметний, так і прикладної підходи. У загальному випадку предметний підхід використовується для побудови початкової інформаційної структури, а прикладної для її вдосконалення з метою підвищення ефективності обробки даних.

При проектуванні інформаційної системи необхідно провести аналіз цілей цієї системи і виявити вимоги до неї окремих користувачів (співробітників організації). Збір даних починається з вивчення сутностей організації і процесів, що використовують ці сутності (докладніше в додатку Б). Сутності групуються по "подібністю" (частоті їх використання для виконання тих чи інших дій) і за кількістю асоціативних зв'язків між ними (літак, пасажир, викладач, дисципліна, студент, сесія т.д.). Сутності або групи сутностей, що володіють найбільшим подібністю і (або) з найбільшою частотою асоціативних зв'язків об'єднуються в предметні БД. (Нерідко сутності об'єднуються в предметні БД без використання формальних методик по "здоровому глузду".

Основна мета проектування БД це скорочення надмірності збережених даних, а отже, економія обсягу використовуваної пам'яті, зменшення витрат на багаторазові операції оновлення надлишкових копій і усунення можливості виникнення протиріч через зберігання в різних місцях відомостей про одне й те ж об'єкті. Так званий, "чистий" проект БД ("Кожен факт в одному місці") можна створити, використовуючи методологію нормалізаціївідносин. І хоча нормалізація повинна використовуватися на завершальній перевірочної стадії проектування БД, ми почнемо обговорення питань проектування з розгляду причин, які змусили Кодда створити основи теорії нормалізації.

6.3. Етапи розробки бази даних

Метою розробки будь-якої бази даних є зберігання та використання інформації про будь-якої предметної області. Для реалізації цієї мети є наступні інструменти:

1. реляційна модель даних – зручний спосіб представлення даних предметної області.
2. Мова SQL - універсальний спосіб маніпулювання такими даними.

Однак очевидно, що для однієї і тієї ж предметної області реляційні відносини можна спроектувати безліччю різних способів. Наприклад, можна спроектувати кілька відносин з великою кількістю атрибутів, або навпаки, рознести все атрибути по великому числу дрібних відносин. Як визначити, за якими ознаками потрібно поміщати атрибути в ті чи інші відносини?

в даній лекції розглядаються способи "хорошого" або "правильного" проектування реляційних відносин. Спочатку ми обговоримо, що означає "хороші" або "правильні" моделі даних. Потім будуть введені поняття першої, другої і третьої нормальних форм відносин (1НФ, 2НФ, 3НФ) і показано, що "хорошими" є відносини третій нормальній формі.

При розробці бази даних зазвичай виділяється кілька рівнів моделювання, за допомогою яких відбувається перехід від предметної області до конкретної реалізації бази даних засобами конкретної СУБД. Можна виділити такі рівні:

- Сама предметна область
- Модель предметної області
- Логічна модель даних
- Фізична модель даних
- Власне база даних і додатки

Предметна область - це частина реального світу, дані про яку ми хочемо відобразити в базі даних. Наприклад, в якості предметної області можна вибрати бухгалтерію будь-якого підприємства, відділ кадрів, банк, магазин і т.д. Предметна область нескінченна і містить як суттєво важливі поняття і дані, так і малозначні або взагалі не значущі дані. Так, якщо в якості предметної області вибрати облік товарів на складі, то поняття "накладна" і "рахунок-фактура" є суттєво важливими поняттями, а то, що співробітниця, приймаюча накладні, має двох дітей - це для обліку товарів неважливо. Однак, з точки зору відділу кадрів дані про наявність дітей є суттєво важливими. Таким чином, важливість даних залежить від вибору предметної області.

Модель предметної області. Модель предметної області - це наші знання про предметну область. Знання можуть бути як у вигляді неформальних знань в мозку експерта, так і виражені формально за допомогою будь-яких засобів. В якості таких засобів можуть виступати текстові описи предметної області, набори посадових інструкцій, правила ведення справ в компанії і т.п. Досвід показує, що текстовий спосіб представлення моделі предметної області вкрай неефективний. Набагато більш інформативними і корисними при розробці баз даних є описи предметної області, виконані за допомогою спеціалізованих графічних нотацій.

Є велика кількість методик опису предметної області. З найбільш відомих можна назвати методику структурного аналізу SADT і засновану на ньому IDEF0, діаграми

потоків даних Гейне-Сарсона, методику об'єктно-орієнтованого аналізу UML, і ін. Модель предметної області описує швидше процеси, що відбуваються в предметній області і дані, використовувані цими процесами. Від того, наскільки правильно змодельована предметна область, залежить успіх подальшої розробки додатків.

Логічна модель даних. На наступному, більш низькому рівні знаходиться логічна модель даних предметної області. Логічна модель описує поняття предметної області, їх взаємозв'язок, а також обмеження на дані, що накладаються предметною областю. Приклади понять - "співробітник", "відділ", "проект", "зарплата". Приклади взаємозв'язків між поняттями - "співробітник числиться рівно в одному відділі", "співробітник може виконувати кілька проектів", "над одним проектом може працювати кілька співробітників". Приклади обмежень - "вік співробітника не менше 16 і не більше 60 років".

Логічна модель даних є початковим прототипом майбутньої бази даних. Логічна модель будується в термінах інформаційних одиниць, але *без прив'язки до конкретної СУБД.* Більш того, логічна модель даних необов'язково повинна бути виражена засобами саме *реляційної* моделі даних. Основним засобом розробки логічної моделі даних в даний момент є різні варіанти *ER-діаграм (Entity-Relationship, діаграми сутність-зв'язок).* Одну і ту ж ER-модель можна перетворити як в реляційну модель даних, так і в модель даних для ієрархічних і мережевих СУБД, або в постреляціонних модель даних. Однак, тому що ми розглядаємо саме реляційні СУБД, то можна вважати, що логічна модель даних для нас формулюється в термінах реляційної моделі даних.

Рішення, прийняті на попередньому рівні, при розробці моделі предметної області, визначають деякі межі, в межах яких можна розвивати логічну модель даних, в межах же цих кордонів можна приймати різні рішення. Наприклад, модель предметної області складського обліку містить поняття "склад", "накладна", "товар". При розробці відповідної реляційної моделі ці терміни обов'язково повинні бути використані, але різних способів реалізації тут багато - можна створити одне ставлення, в якому будуть присутні в якості атрибутів "склад", "накладна", "товар", а можна створити три окремих відносини, по одному на кожне поняття.

При розробці логічної моделі даних виникають питання: чи добре спроектовані відносини? Чи правильно вони відображають модель предметної області, а отже і саму предметну область?

Фізична модель даних. На ще більш низькому рівні знаходиться фізична модель даних. Фізична модель даних описує дані засобами конкретної СУБД. Ми будемо вважати, що фізична модель даних реалізована засобами саме *реляційної* СУБД, хоча, як вже сказано вище, це необов'язково. Відносини, розроблені на стадії формування логічної моделі даних, перетворюються в таблиці, атрибути стають стовпцями таблиць, для ключових атрибутів створюються унікальні індекси, домени перетворюються в типи даних, прийняті в конкретній СУБД. Обмеження наявні в логічної моделі даних, реалізуються різними засобами СУБД, наприклад, за допомогою індексів, декларативних обмежень цілісності, тригерів, збережених процедур. При цьому знову-таки рішення, прийняті на рівні логічного моделювання визначають деякі межі, в межах яких можна розвивати фізичну модель даних. Точно також, в межах цих кордонів можна приймати різні рішення. Наприклад, відносини, що містяться в логічної моделі даних, повинні бути перетворені в таблиці, але для кожної таблиці можна додатково оголосити різні індекси, що підвищують швидкість доступу до даних.

При розробці фізичної моделі даних виникають питання: чи добре спроектовані таблиці? Чи правильно вибрані індекси? Наскільки багато програмного коду у вигляді тригерів і збережених процедур необхідно розробити для підтримки цілісності даних?

Власне база даних і додатки. І, нарешті, як результат попередніх етапів з'являється власне сама база даних. База даних реалізована у конкретній програмно-апаратної основі, і вибір цієї основи дозволяє істотно підвищити швидкість роботи з базою даних. Наприклад,

можна вибирати різні типи комп'ютерів, змінювати кількість процесорів, обсяг оперативної пам'яті, дискові підсистеми і т.п. Дуже велике значення має також настройка СУБД в межах обраної програмно-апаратної платформи.
Але знову рішення, прийняті на попередньому рівні - рівні фізичного проектування, визначають межі, в межах яких можна приймати рішення щодо вибору програмно-апаратної платформи і налаштування СУБД.
Таким чином ясно, що рішення, прийняті на кожному етапі моделювання і розробки бази даних, будуть позначатися на подальших етапах. Тому особливу роль відіграє прийняття правильних рішень *на ранніх етапах моделювання.*

6.4. Критерії оцінки якості логічної моделі даних

Мета даної лекції - описати деякі принципи побудови *хороших логічних моделей даних.* Хороших в тому сенсі, що рішення, прийняті в процесі логічного проектування приводили б до хорошим фізичним моделям і в кінцевому підсумку до хорошої роботи бази даних.
Для того щоб оцінити якість прийнятих рішень на рівні логічної моделі даних, необхідно сформулювати деякі критерії якості в термінах фізичної моделі і конкретної реалізації і подивитися, як різні рішення, прийняті в процесі *логічного* моделювання, впливають на якість *фізичної* моделі і на швидкість роботи бази даних.
Звичайно, таких критеріїв може бути дуже багато і вибір їх в достатній мірі довільний. Ми розглянемо деякі з таких критеріїв, які є безумовно важливими з точки зору отримання якісної бази даних:

- Адекватність бази даних предметної області.
- Легкість розробки і супроводу бази даних.
- Швидкість виконання операцій оновлення даних (вставка, оновлення, видалення кортежів).
- Швидкість виконання операцій вибірки даних.

Адекватність бази даних предметної області
База даних повинна адекватно відображати предметну область. Це означає, що повинні бути виконані такі умови:

1. Стан бази даних в кожен момент часу має відповідати стану предметної області.
2. зміна стану предметної області має приводити до відповідної зміни стану бази даних
3. Обмеження предметної області, відображені в моделі предметної області, повинні деяким чином відбиватися і враховуватися базі даних.

Практично будь-яка база даних, за винятком зовсім елементарних, містить певну кількість програмного коду у вигляді тригерів і збережених процедур.
Збережені процедури - це процедури і функції, що зберігаються
безпосередньо в базі даних в відкомпілювався вигляді і які можуть запускатися
користувачами або додатками, що працюють з базою даних. Збережені процедури зазвичай пишуться або на спеціальному процедурному розширенні мови SQL (наприклад, PL / SQL для ORACLE або Transact-SQL для MS SQL Server), або на
деякому універсальному мові програмування, наприклад, C ++, з включенням в код операторів SQL у відповідності зі спеціальними правилами такого включення.
Основне призначення процедур - реалізація бізнес-процесів предметної області.
Тригери - це збережені процедури, пов'язані з деякими подіями, що відбуваються під час роботи бази даних. В якості таких подій виступають операції вставки, оновлення та видалення рядків таблиць. Якщо в базі даних визначено певний тригер, то він запускається *автоматично* завжди при виникненні події, з яким цей тригер пов'язаний. Дуже важливим є те, що користувач не може обійти тригер. Тригер спрацьовує незалежно від того, хто з користувачів і яким способом ініціював подія, яка викликала запуск тригера.

Таким чином, основне призначення тригерів - автоматична підтримка цілісності бази даних. Тригери можуть бути як досить простими, наприклад, підтримують кількість посилань цілісність, так і досить складними, що реалізують будь-які складні обмеження предметної області або складні дії, які повинні відбутися при настанні деяких подій. Наприклад, з операцією вставки нового товару накладну може бути пов'язаний тригер, який виконує наступні дії - перевіряє, чи є необхідна кількість товару, при наявності товару додає його в накладну і зменшує дані про наявність товару на складі, при відсутності товару формує замовлення на поставку відсутнього товару і тут же посилає замовлення по електронній пошті постачальнику.

Очевидно, що чим більше програмного коду у вигляді тригерів і збережених процедур містить база даних, тим складніше її розробка і подальший супровід.

Швидкість операцій оновлення даних (вставка, оновлення, видалення)

На рівні логічного моделювання ми визначаємо реляційні відносини і атрибути цих відносин. На цьому рівні ми не можемо визначати будь-які фізичні структури зберігання (індекси, хешування і т.п.). Єдине, чим ми можемо управляти - розподілом атрибутів з різних відносин. Можна описати мало відносин з великою кількістю атрибутів, або багато відносин, кожне з яких містить мало атрибутів. Таким чином, необхідно спробувати відповісти на питання - чи впливає кількість відносин і кількість атрибутів у відносинах на швидкість виконання операцій оновлення даних.

Таке питання, звичайно, не є достатньо коректним, тому що швидкість виконання операцій з базою даних сильно залежить від фізичної реалізації бази даних. Проте, спробуємо *якісно* оцінити цей вплив при *однакові підходи до фізичного моделювання.*

Основними операціями, що змінюють стан бази даних, є операції вставки, оновлення та видалення записів. У базах даних, які потребують постійних змін (складський облік, системи продажів квитків і т.п.) продуктивність визначається швидкістю виконання великої кількості невеликих операцій вставки, оновлення та видалення.

Розглянемо операцію вставки запису в таблицю. Вставка запису провадиться одну з вільних сторінок пам'яті, виділеної для даної таблиці. СУБД постійно зберігає інформацію про наявність і розташування вільних сторінок. Якщо для таблиці не створені індекси, то операція вставки виконується фактично з однаковою швидкістю незалежно від розміру таблиці і від кількості атрибутів в таблиці. Якщо в таблиці є індекси, то при виконанні операції вставки записи індекси повинні бути перебудовані. Таким чином, швидкість виконання операції вставки *зменшується при збільшенні кількості індексів* у таблиці і *мало залежить від числа рядків* в таблиці.

Розглянемо операції оновлення та видалення записів з таблиці. Перш, ніж оновити або видалити запис, її необхідно знайти. Якщо таблиці не індексована, то єдиним способом пошуку є послідовне сканування таблиці в пошуку потрібного запису. В цьому випадку, швидкість операцій оновлення і видалення істотно збільшується зі збільшенням кількості записів в таблиці і не залежить від кількості атрибутів. Але насправді неіндексовані таблиці практично ніколи не використовуються. Для кожної таблиці зазвичай оголошується один або кілька індексів, відповідний потенційним джерелам. За допомогою цих індексів пошук запису виробляється дуже швидко і практично не залежить від кількості рядків і атрибутів в таблиці (хоча, звичайно, деяка залежність є). Якщо для таблиці оголошено кілька індексів, то при виконанні операцій оновлення і видалення ці індекси повинні бути перебудовані, на що витрачається додатковий час. Таким чином, швидкість виконання операцій оновлення і видалення також *зменшується при збільшенні кількості індексів* у таблиці і *мало залежить від числа рядків* в таблиці.

Можна припустити, що чим більше атрибутів має таблиця, тим більше для неї буде оголошено індексів. Ця залежність, звичайно, не пряма, але *при однакових підходах* до фізичного моделювання зазвичай так і відбувається. Таким чином, можна прийняти допущення, що *чим більше атрибутів мають відношення,* розроблені в ході логічного

моделювання, *тим повільніше будуть виконуватися операції оновлення даних,* за рахунок витрати часу на перебудову більшої кількості індексів.
Додаткові міркування на користь наведеної тези про уповільнення виконання операцій оновлення даних (вплив журналізації, довжини рядків таблиць) наведені в роботі А.Прохорова.
Швидкість операцій вибірки даних
Одне з призначень бази даних - надання інформації користувачам. Інформація витягується з реляційної бази даних за допомогою оператора SQL - SELECT. Однією з найбільш дорогих операцій при виконанні оператора SELECT є операція з'єднання таблиць. Таким чином, чим більше взаємопов'язаних відносин було створено в ході логічного моделювання, тим більша ймовірність того, що при виконанні запитів ці відносини будуть з'єднуватися, і, отже, тим повільніше будуть виконуватися запити. Таким чином, збільшення кількості відносин призводить до уповільнення виконання операцій вибірки даних, особливо, якщо запити заздалегідь невідомі.

Питання для самоконтролю:

1. Які ви знаєте типи зв'язків між сутностями?
2. Що ви розумієте під терміном "Первинний ключ"?
3. Наведіть приклад потрійного складеного ключа.
4. Поясніть особливості поняття "Зовнішній ключ".
5. Поясніть зміст поняття "Схема реляційної БД".
6. Назвіть основні етапи проектування бази даних.

Лекція 7.

Нормалізація відносин

1. Перша нормальна форма (1НФ).
2. Аномалії реляційного відносини.
3. Функціональні залежності.

Розглянемо в якості предметної області деяку організацію, яка виконує деякі проекти. Модель предметної області опишемо наступним неформальним текстом:

1. Співробітники організації виконують проекти.
2. Проекти складаються з декількох завдань.
3. Кожен співробітник може брати участь в одному або декількох проектах, або тимчасово не брати участь ні в яких проектах.
4. Над кожним проектом може працювати кілька співробітників, або тимчасово проект може бути призупинений, тоді над ним не працює жоден співробітник.
5. Над кожним завданням в проекті працює рівно один співробітник.
6. Кожен співробітник числиться в одному відділі.
7. Кожен співробітник має телефон, що знаходиться у відділі співробітника.

В ході додаткового уточнення того, які дані необхідно враховувати, з'ясувалося наступне:

1. Про кожного співробітника необхідно зберігати табельний номер і прізвище. Табельний номер є унікальним для кожного співробітника.

Кожен відділ має унікальний номер.
Кожен проект має номер і найменування. Номер проекту є унікальним. Кожна робота з проекту має номер, унікальний в межах проекту. Роботи в різних проектах можуть мати однакові номери.
1НФ (Перша Нормальна Форма)
Поняття першої нормальної форми вже обговорювалося. *Перша нормальна форма (1НФ)* - це звичайне відношення. Відповідно до нашого визначення відносин, будь-яке відношення

автоматично вже знаходиться в 1НФ. Нагадаємо коротко властивості відносин (це і будуть властивості 1НФ):

- У відношенні немає однакових кортежів.
- Кортежі не впорядковані.
- Атрибути не впорядковані і розрізняються по найменуванню.
- Всі значення атрибутів атомарний.

В ході логічного моделювання на першому кроці запропоновано зберігати дані в одному відношенні, що має такі атрибути:

СОТРУДНІКІ_ОТДЕЛИ_ПРОЕКТИ (Н_СОТР, ФАМ, Н_ОТД, ТЕЛ, Н_ПРО, ПРОЕКТ, Н_ЗАДАН)

де

Н_СОТР - табельний номер співробітника

ФАМ - прізвище співробітника

Н_ОТД - номер відділу, в якому значиться співробітник

ТЕЛ - телефон співробітника

Н_ПРО - номер проекту, над яким працює співробітник

ПРОЕКТ - найменування проекту, над яким працює співробітник

Н_ЗАДАН - номер завдання, над яким працює співробітник Оскільки кожен співробітник в кожному проекті виконує рівно одне завдання, то в якості потенційного ключа відносини необхідно взяти пару атрибутів { Н_СОТР, Н_ПРО}.

У поточний момент стан предметної області відображається такими фактами:

• Співробітник Іванов, який працює в 1 відділі, виконує в першому проекті "Космос" завдання 1 і в другому проекті "Клімат" завдання 1.

• Співробітник Петров, який працює в 1 відділі, виконує в першому проекті "Космос" завдання 2.

• Співробітник Сидоров, який працює у 2 відділі, виконує в першому проекті "Космос" завдання 3 і в другому проекті "Клімат" завдання 2.

Цей стан відбивається в таблиці (курсивом виділено ключові атрибути):

Таблиця 7.1.

Н_СОТР	ФАМ	Н_ОТД	ТЕЛ	Н_ПРО	ПРОЕКТ	Н_ЗАДАН
1	Иванов	1	11-22-33	1	Космос	1
1	Иванов	1	11-22-33	2	Климат	1
2	Петров	1	11-22-33	1	Космос	2
3	Сидоров	2	33-22-11	1	Космос	3
3	Сидоров	2	33-22-11	2	Климат	2

2. Аномалії в реляційних видношеннях

Навіть одного погляду на таблиці відношення СОТРУДНІКІ_ОТДЕЛИ_ПРОЕКТИ досить, щоб побачити, що дані зберігаються в ній *великою надмірністю.* У багатьох рядках повторюються прізвища співробітників, номери телефонів, найменування проектів. Крім того, в даному відношенні зберігаються разом незалежні один від одного дані - і дані про співробітників, і про відділи, і про проекти, і про роботи по проектам. Поки ніяких дій з відношенням не проводиться, це не страшно. Але як тільки стан предметної області змінюється, то, при спробах відповідним чином змінити стан бази даних, виникає велика кількість проблем.

Історично ці проблеми отримали назву *аномалії оновлення.* Спроби дати суворе поняття аномалії в базі даних не є цілком задовільними. У даних роботах аномалії визначені як протиріччя між моделлю предметної області і фізичною моделлю даних, підтримуваних засобами конкретної СУБД. "Аномалії виникають в тому випадку, коли наші знання про предметну області виявляються, по якимось причин, невимовними в схемі БД або входять в протиріччя з нею ". Ми дотримуємося іншої точки зору, що полягає в тому, що аномалій

в сенсі визначень згаданих авторів немає, а є або *неадекватність моделі даних* предметної області, або деякі *додаткові труднощі в реалізації обмежень* предметної області засобами СУБД. Більш глибоке обговорення проблеми строгого визначення поняття аномалій виходить за межі даної роботи.

Таким чином, ми будемо дотримуватися інтуїтивного поняття аномалії як неадекватності моделі даних предметної області, (що говорить насправді про те, що логічна модель даних просто невірна!) Або як необхідність додаткових зусиль для реалізації всіх обмежень визначених у предметної області (додатковий програмний код у вигляді тригерів або збережених процедур).

Оскільки аномалії виявляють себе при виконанні операцій, що змінюють стан бази даних, то розрізняють наступні види аномалій:

Аномалії вставки (INSERT).

Аномалії оновлення (UPDATE).

Аномалії видалення (DELETE).

У відносинах СОТРУДНІКІ_ОТДЕЛИ_ПРОЕКТИ можна навести приклади таких аномалій:

Аномалії вставки (INSERT)

у відношення СОТРУДНІКІ_ОТДЕЛИ_ПРОЕКТИ не можна вставити дані про співробітника, який поки не бере участь ні в одному проекті. Дійсно, якщо, наприклад, у другому відділі з'являється новий співробітник, скажімо, Пушник, і він поки не бере участь ні в одному проекті, то ми повинні вставити у відношення кортеж (4, Пушник, 2, 33-22-11, null, null , null). Це зробити неможливо, тому що атрибут Н_ПРО (Номер проекту) входить до складу потенційного ключа, і, отже, не може містити null-значень.

Також можна вставити дані про проект, над яким поки не працює жоден співробітник.

Причина аномалії - зберігання в одному відношенні різнорідної інформації (і про співробітників, і про проекти, і про роботи по проекту).

Висновок - логічна модель даних неадекватна моделі предметної області.

База даних, заснована на такій моделі, працюватиме неправильно.

Аномалії оновлення (UPDATE)

Прізвища співробітників, найменування проектів, номери телефонів повторюються в багатьох кортежі відносини. Тому якщо співробітник змінює прізвище, або проект змінює найменування, або змінюється номер телефону, то такі зміни необхідно *одночасно* виконати у всіх місцях, де це прізвище, найменування або номер телефону зустрічаються, інакше ставлення стане некоректним (наприклад, один і той же проект в різних кортежі буде називатися по-різному).

Таким чином, оновлення бази даних одним дією реалізувати неможливо. Для підтримки відносини в цілісному стані необхідно написати тригер, який при оновленні запису коректно виправляв би дані і в інших місцях.

Причина аномалії - надмірність даних, також породжена тим, що в одному відношенні зберігається різнорідна інформація.

Висновок - збільшується складність розробки бази даних. База даних, заснована на такій моделі, працюватиме правильно тільки при наявності додаткового програмного коду у вигляді тригерів.

Аномалії видалення (DELETE)

При видаленні деяких даних може відбутися втрата іншої інформації. Наприклад, якщо закрити проект "Космос" і видалити всі рядки, в яких він зустрічається, то будуть втрачені всі дані про співробітника Петрові. Якщо видалити співробітника Сидорова, то буде втрачена інформація про те, що у відділі номер 2 знаходиться телефон 33- 22-11. Якщо за проектом тимчасово припинені роботи, то при видаленні даних про роботи за цим проектом будуть видалені і дані про сам проект (найменування проекту). При цьому якщо був співробітник, який працював тільки над цим проектом, то будуть втрачені і дані про це працівника.

Причина аномалії - зберігання в одному відношенні різнорідної інформації (і про співробітників, і про проекти, і про роботи по проекту).
Висновок - логічна модель даних неадекватна моделі предметної області.
База даних, заснована на такій моделі, працюватиме неправильно.

3. Функціональні залежності
Відношення СОТРУДНІКІ_ОТДЕЛИ_ПРОЕКТИ знаходиться в 1НФ, при цьому, як було показано вище, логічна модель даних не адекватна моделі предметної області. Таким чином, першою нормальної форми *недостатньо* для правильного моделювання даних.
Для усунення зазначених аномалій (а насправді *для правильного проектування моделі даних!)* застосовується метод нормалізації відносин. Нормалізація заснована на понятті функціональної залежності атрибутів відносини.
Визначення 8.1. нехай *R* - відношення. безліч атрибутів *Y функціонально залежно* від безлічі атрибутів *X (функціонально визначає Y)* тоді і тільки тоді, коли *для будь-якого стану* відносини *R* для будь-яких кортежів $r, r \in R$ того, що $r_1 X = r_2 X$ випливає, що $r_1 Y = r_2 Y$. (Тобто у всіх кортежі, що мають однакові значення атрибутів *X,* значення атрибутів *Y* також збігаються *в будь-якому стані* відносини *R).* Символічно функціональна залежність записується $X -> Y$.
Безліч атрибутів *X* називається *детермінантою функціональної залежності,* а безліч атрибутів *Y* називається *залежною частиною.*
Зауваження. якщо атрибути *X* складають потенційний ключ відносини *R,* то *будь-який* атрибут відносини *R* функціонально залежить від *X.*
Приклад в відношенні СОТРУДНІКІ_ОТДЕЛИ_ПРОЕКТИ можна навести такі приклади функціональної залежності:
Залежність атрибутів від ключа відносини:
{Н_СОТР, Н_ПРО} - >ФАМ
{Н_СОТР, Н_ПРО} - > Н_ОТД
{Н_СОТР, Н_ПРО} - >ТЕЛ
{Н_СОТР, Н_ПРО} - >ПРОЕКТ
{Н_СОТР, Н_ПРО} - >Н_ЗАДАН

Залежність атрибутів, характеризують співробітника від табельного номера співробітника:
Н_СОТР - >ФАМ
Н_СОТР - >Н_ОТД
Н_СОТР - >ТЕЛ
Залежність найменування проекту від номера проекту:
Н_ПРО - >ПРОЕКТ
Залежність номера телефону від номера відділу:
Н_ОТД - >ТЕЛ
Зауваження. Наведені функціональні залежності *не виведено* з зовнішнього вигляду відносини, наведеного в таблиці 1. Ці залежності відображають взаємозв'язки, виявлені між об'єктами предметної області і є додатковими обмеженнями, обумовленими предметною областю. Таким чином, функціональна залежність - *семантичне поняття.* Вона виникає, коли за значеннями одних даних в предметної області можна визначити значення інших даних. Наприклад, знаючи табельний номер співробітника, можна визначити його прізвище, по номеру відділу можна визначити телефону. функціональна залежність *задає додаткові обмеження* на дані, які можуть зберігатися в стосунках. Для коректності бази даних (адекватності предметної області) необхідно при виконанні операцій модифікації бази даних перевіряти всі обмеження, задані певні функціональними залежностями.
Функціональні залежності відносин і математичне поняття функціональної залежності

Функціональна залежність атрибутів відносини нагадує поняття функціональної залежності в математиці. *Але це не одне і те ж.* Для порівняння нагадаємо математичне поняття функціональної залежності:
Функціональна залежність (*функція)* - це трійка об'єктів
{X, Y, f}, де
X - безліч (*область визначення),*
Y - безліч (*безліч значень),*
f - правило, згідно з яким кожному елементу $x \in X$ ставиться у відповідність один і тільки один елемент $y \in Y$(*правило функціональної залежності).*
Функціональна залежність зазвичай позначається як $f : X -> Y$ *або* $y = f(x)$
Зауваження. правило *f* може бути задано будь-яким способом - у вигляді формули (найчастіше), за допомогою таблиці значень, за допомогою графіка, текстовим описом і т.д.
Функціональна залежність атрибутів відносини теж *нагадує* це визначення. дійсно:

- Як області визначення виступає домен, на якому визначено атрибут *X (* або декартовій твір доменів, якщо *X* є безліччю атрибутів)
- Як безлічі значень виступає домен, на якому визначено атрибут *Y (* або декартовій твір доменів)
- правило *f* реалізується наступним алгоритмом - 1) за даним значенням атрибута *X* знайти *будь-який* кортеж відносини, що містить це значення, 2) значення атрибута *Y* в цьому кортежі і буде значенням функціональної залежності, відповідним даному *X.* Визначення функціональної залежності щодо гарантує, що знайдене значення *Y не залежить від вибору кортежу,* тому правило *f* визначено коректно.

Відмінність від математичного поняття відносини полягає в тому, що, якщо розглядати математичне поняття функції, то *для фіксованого значення* $x \in X$ відповідне значення функції y= f (x) *завжди одне і те ж.* Наприклад, якщо задана функція $y = x^2$ то для значення x=*2* відповідне значення *y завжди* дорівнюватиме 4.
В противагу цьому в відношеннях значення залежного атрибута може приймати *різні* значення в різних станах бази даних.
Наприклад, атрибут ФАМ функціонально залежить від атрибута Н_СОТР. Припустимо, що зараз співробітник з табельною номером 1 має прізвище Іванов, тобто при значенні детермінанта рівного 1, значення залежного аргументу одно "Іванов". Але співробітник може змінити прізвище, наприклад на "Сидоров". Тепер *при тому ж* значенні детермінанта, рівного 1, значення залежного аргументу одно "Сидоров".
Таким чином, поняття функціональної залежності атрибутів можна вважати повністю еквівалентним математичному поняттю функціональної залежності, тому що значення цієї залежності різні при різних станах відносини, і, найголовніше, ці значення можуть змінюватися *непередбачувано.*
Функціональна залежність атрибутів стверджує лише те, що для кожного конкретного стану бази даних за значенням одного атрибута (детермінанта) можна однозначно визначити значення іншого атрибута (залежної частини). Але конкретні значення залежної частини можуть бути різні в різних станах бази даних.

Питання для самоконтролю:

1. Дайте визначення поняттю "Універсальне відношення"?
2. Що ви розумієте під терміном "Аномалії відносини"?
3. Наведіть приклад ставлення в першій нормальній формі.
4. Поясніть особливості поняття "Функціонально-залежні атрибути ".
5. Назвіть основні етапи нормалізації відносини до першої нормальної форми.

Лекція 7.

НОРМАЛІЗАЦІЯ ВІДНОСИН (частина 2)

1. Друга нормальна форма (2 НФ).
2. Третя нормальна форма (3 НФ).
3. Алгоритм нормалізації: приведення відношення до 3НФ
4. Порівняння нормалізованих і ненормалізованих моделей

1. Друга нормальна форма (2НФ)

Визначення 9.1 .. Відношення знаходиться в *другий нормальної формі (2НФ)* тоді і тільки тоді, коли відношення знаходиться в 1НФ і *немає неключових атрибутів, залежних від частини складного ключа.*

(*неключових атрибут* - це атрибут, який не входить до складу жодного потенційного ключа).

Зауваження. Якщо потенційний ключ відносини є простим, то ставлення автоматично перебуває в 2НФ.

ставлення СОТРУДНІКІ_ОТДЕЛИ_ПРОЕКТИ НЕ знаходиться в 2НФ, тому що є атрибути, які залежать від частини складного ключа:

Залежність атрибутів, характеризують співробітника від табельного номера співробітника є залежністю від частини складного ключа:

Н_СОТР ->ФАМ
Н_СОТР ->Н_ОТД
Н_СОТР ->ТЕЛ

Залежність найменування проекту від номера проекту є залежністю від частини складного ключа:

Н_ПРО ПРОЕКТ

Для того, щоб усунути залежність атрибутів від частини складного ключа, потрібно зробити *декомпозицію* відносини на кілька відносин. При цьому *ті атрибути, які залежать від частини складного ключа,* виносяться в окреме відношення.

Відношення СОТРУДНІКІ_ОТДЕЛИ_ПРОЕКТИ декомпозіруем на три відносини - СОТРУДНІКІ_ОТДЕЛИ, ПРОЕКТИ, ЗАВДАННЯ.

Відношення СОТРУДНІКІ_ОТДЕЛИ (Н_СОТР, ФАМ, Н_ОТД, ТЕЛ):

Функціональні залежності:

Залежність атрибутів, характеризують співробітника від табельного номера співробітника:

Н_СОТР ->ФАМ
Н_СОТР ->Н_ОТД
Н_СОТР ->ТЕЛ

Залежність номера телефону від номера відділу:

Н_ОТД ->ТЕЛ

Н_СОТР	ФАМ	Н_ОТД	ТЕЛ
1	Иванов	1	11-22-33
2	Петров	1	11-22-33
3	Сидоров	2	33-22-11

Відношення СОТРУДНИКИ_ОТДЕЛЫ.

Відношення ПРОЕКТИ (Н_ПРО, ПРОЕКТ):

Функціональні залежності:

Н_ПРО ->ПРОЕКТ

Н_ПРО	ПРОЕКТ
1	Космос
2	Климат

Відношення ПРОЕКТИ.
Відношення ЗАВДАННЯ (Н_СОТР, Н_ПРО, Н_ЗАДАН):
Функціональні залежності:
{Н_СОТР, Н_ПРО} Н_ЗАДАН

$\rightarrow$

Н_СОТР	Н_ПРО	Н_ЗАДАН
1	1	1
1	2	1
2	1	2
3	1	3
3	2	2

Відношення ЗАВДАННЯ.
Аналіз декомпозіровать відносин
Відношення, отримані в результаті декомпозиції, знаходяться в 2НФ. дійсно, відносини СОТРУДНІКІ_ОТДЕЛИ і ПРОЕКТИ мають прості ключі, отже автоматично знаходяться в 2НФ, ставлення ЗАВДАННЯ має складний ключ, але єдиний неключових атрибут Н_ЗАДАН функціонально залежить від усього ключа {Н_СОТР, Н_ПРО}.
Частина аномалій поновлення усунена. Так, дані про співробітників і проектах тепер зберігаються в різних відносинах, тому при появі співробітників, які не беруть участі ні в одному проекті просто додаються кортежі в відношення СОТРУДНІКІ_ОТДЕЛ Точно також, при появі проекту, над яким не працює жоден співробітник, просто вставляється кортеж в ставлення ПРОЕКТИ.
Прізвища співробітників і найменування проектів тепер зберігаються без надмірності. Якщо співробітник змінить прізвище або проект змінить назву, то таке оновлення буде вироблено в одному місці.
Якщо за проектом тимчасово припинені роботи, але потрібно, щоб сам
проект зберігся, то для цього проекту видаляються відповідні кортежі щодо ЗАВДАННЯ, дані про сам проект і дані про співробітників, які брали участь у проекті, залишаються в стосунках ПРОЕКТИ і СОТРУДНІКІ_ОТДЕЛИ.
Проте, частина аномалій дозволити не вдалося.
Решта аномалії вставки (INSERT)

У відношення СОТРУДНІКІ_ОТДЕЛИ не можна вставити кортеж (4, Пушник, 1, 33-22-11), тому що при цьому вийде, що два співробітника з 1-го відділу (Іванов і Пушник) мають різні номери телефонів, а це суперечить моделі предметної області.

У цій ситуації можна запропонувати два рішення, в залежності від того, що реально відбулося в предметної області. Інший номер телефону може бути введений з двох причин - по помилку людини, що вводить дані про нового співробітника, або тому що номер у відділі дійсно змінився. Тоді можна написати тригер, який при вставці запису про співробітника перевіряє, чи збігається телефон з уже наявними телефоном у іншого співробітника цього ж відділу.
Якщо номери відрізняються, то система повинна задати питання, чи залишити старий номер у відділі або замінити його новим. Якщо потрібно залишити старий номер (новий номер введений помилково), то кортеж з даними про нового співробітника буде вставлений, але номер телефону буде у нього буде той, який вже є в відділі (в даному випадку, 11-22-33). Якщо ж номер в відділі дійсно змінився, то кортеж буде вставлений з

новим номером, і одночасно будуть змінені номери телефонів у всіх співробітників цього ж відділу. І в тому і в іншому випадку не обійтися без розробки громіздкого тригера.
Причина аномалії - надмірність даних, породжена тим, що в одному відношенні зберігається різнорідна інформація (про співробітників і про відділи).
Висновок - збільшується складність розробки бази даних. База даних, заснована на такій моделі, працюватиме правильно тільки при наявності додаткового програмного коду у вигляді тригерів.
Решта аномалії оновлення (UPDATE)
Одні і ті ж номери телефонів повторюються в багатьох кортежі відносини. Тому якщо у відділі змінюється номер телефону, то такі зміни необхідно одночасно виконати у всіх місцях, де цей номер телефону зустрічаються,

інакше ставлення стане некоректним. Таким чином, оновлення бази даних одним дією реалізувати неможливо. Необхідно написати тригер, який при оновленні запису коректно виправляє номери телефонів в інших місцях.
Причина аномалії - надмірність даних, також породжена тим, що в одному відношенні зберігається різнорідна інформація.
Висновок - збільшується складність розробки бази даних. База даних, заснована на такій моделі, працюватиме правильно тільки при наявності додаткового програмного коду у вигляді тригерів.
Решта аномалії видалення (DELETE)
При видаленні деяких даних як і раніше може відбутися втрата іншої інформації. Наприклад, якщо видалити співробітника Сидорова, то буде втрачена інформація про те, що у відділі номер 2 знаходиться телефон 33-22-11.
Причина аномалії - зберігання в одному відношенні різнорідної інформації (і про співробітників, і про відділи).
Висновок - логічна модель даних неадекватна моделі предметної області.
База даних, заснована на такій моделі, працюватиме неправильно.
Зауважимо, що при переході до другої нормальної форми відносини стали *майже* адекватними предметної області. Залишилися також труднощі в розробці бази даних, пов'язані з необхідністю написання тригерів, що підтримують цілісність бази даних. Ці труднощі тепер пов'язані тільки з одним відношенням СОТРУДНІКІ_ОТДЕЛИ.
2. Третя нормальна форма
Визначення атрибути називаються *взаємно незалежними,* якщо жоден з них не є функціонально залежним від іншого.
Визначення ставлення *R* знаходиться в *третій нормальній формі (3НФ)* тоді і тільки тоді, коли відношення знаходиться в 2НФ і *все неключових атрибути взаємно незалежні.*
Відношення СОТРУДНІКІ_ОТДЕЛИ не перебуває у 3НФ, тому що є функціональназалежність неключових атрибутів (Залежність номера телефону від номера відділу):
Н_ОТД ->ТЕЛ $\rightarrow$
Для того, щоб усунути залежність неключових атрибутів, потрібно зробити декомпозицію відносини на кілька відносин. При цьому *ті неключових атрибутів, які є залежними,* виносяться в окреме відношення.
Відношення СОТРУДНІКІ_ОТДЕЛИ декомпозіруем на два відносини - СПІВРОБІТНИКИ, ВІДДІЛИ.
Відношення СПІВРОБІТНИКИ (Н_СОТР, ФАМ, Н_ОТД):
Функціональні залежності:
Залежність атрибутів, характеризують співробітника від табельного номера співробітника:
Н_СОТР ->ФАМ $\rightarrow$
Н_СОТР ->Н_ОТД $\rightarrow$
Н_СОТР ->ТЕЛ

Н_СОТР	ФАМ	Н_ОТД
1	Иванов	1
2	Петров	1
3	Сидоров	2

Відношення СПІВРОБІТНИКИ.
Відношення ВІДДІЛИ (Н_ОТД, ТЕЛ):
Функціональні залежності:
Залежність номера телефону від номера відділу:

Н_ОТД ->ТЕЛ

Н_ОТД	ТЕЛ
1	11-22-33
2	33-22-11

Відношення ВІДДІЛИ.
Звернемо увагу на те, що атрибут Н_ОТД, *що не був ключовим* у відносинах СОТРУДНІКІ_ *стає потенційним ключем* у відносинах ВІДДІЛИ. Саме за рахунок цього усувається надмірність, пов'язана з багаторазовим зберіганням одних і тих же номерів телефонів.
Висновок. Таким чином, всі виявлені аномалії поновлення усунені. Реляційна
модель, що складається з чотирьох відносин
СПІВРОБІТНИКИ, ВІДДІЛИ, ПРОЕКТИ, ЗАВДАННЯ, знаходяться в третій нормальній формі, є адекватною описаної моделі предметної області, і вимагає наявності тільки тих тригерів, які підтримують кількість посилань цілісність. Такі тригери є стандартними і не вимагають великих зусиль в розробці.
3. Алгоритм нормалізації: приведення відношення до 3НФ Отже, алгоритм нормалізації (тобто алгоритм приведення відносин до 3НФ) описується наступним чином.
Крок 1 (Приведення до 1НФ). На першому кроці задається одне або кілька відносин, що відображають поняття предметної області. За моделлю предметної області (не за зовнішнім виглядом отриманих відносин!) виписуються виявлені функціональні залежності. Всі відносини автоматично знаходяться в 1НФ.
Крок 2 (Приведення до 2НФ). Якщо в деяких відносинах виявлена залежність атрибутів від частини складного ключа, то проводимо декомпозицію цих відносин на кілька відносин в такий спосіб: ті атрибути, які залежать від частини складного ключа виносяться в окреме відношення разом з цією частиною ключа. У вихідному відношенні залишаються всі ключові атрибути.
Крок 3 (Приведення до 3НФ). Якщо в деяких відносинах виявлена залежність деяких неключових атрибутів інших неключових атрибутів, то проводимо декомпозицію цих відносин таким чином: ті неключових атрибутів, які залежать інших неключових атрибутів виносяться в окреме відношення. У новому відношенні ключем стає детермінант функціональної залежності.
Зауваження. На практиці, при створенні логічної моделі даних, як правило, не дотримуються прямо наведеним алгоритмом нормалізації. Досвідчені розробники зазвичай відразу будують відносини в 3НФ. Крім того, основним засобом розробки логічних моделей даних є різні варіанти ER-діаграм. Особливість цих діаграм в тому, що вони відразу дозволяють створювати відносини в 3НФ. Проте, наведений алгоритм важливий з двох причин.

По перше, цей алгоритм показує, які проблеми виникають при розробці слабо нормалізованих відносин.
По-друге, як правило, модель предметної області ніколи не буває правильно розроблена з першого кроку. Експерти предметної області можуть забути про будь-що згадати, розробник може неправильно зрозуміти експерта, під час розробки можуть змінитися правила, прийняті в предметної області, і т.д. Все це може привести до появи нових залежностей, які були відсутні в первісної моделі предметної області. Тут як раз і необхідно використовувати алгоритм нормалізації хоча б для того, щоб переконатися, що відносини залишилися в 3НФ і логічна модель не погіршилася. 9.4. Порівняння нормалізованих і ненормалізованих моделей.

4. Порівняння нормалізованих і ненормалізованих моделей

Зберемо воєдино результати аналізу критеріїв, за якими ми хотіли оцінити вплив логічного моделювання даних на якість фізичних моделей даних і продуктивність бази даних. Більш сильно нормалізовані відносини виявляються краще спроектовані (три плюса, один мінус). Вони більше відповідають предметної області, легше в розробці, для них швидше виконуються операції модифікації бази даних. Правда, це досягається ціною деякого уповільнення виконання операцій вибірки даних.
У слабо нормалізованих відносин єдине перевага - якщо до бази даних звертатися тільки до запитів на вибірку даних, то для слабо нормалізованих відносин такі запити виконуються швидше. Це пов'язано тим, що в таких відносинах уже як би вироблено з'єднання відносин і на це не витрачається час при вибірці даних.
Таким чином, вибір ступеня нормалізації відносин залежить від характеру запитів, з якими найчастіше звертаються до бази даних.
Висновки
При розробці бази даних можна виділити кілька рівнів моделювання:

- Сама предметна область
- Модель предметної області
- Логічна модель даних
- Фізична модель даних
- Власне база даних і додатки

Ключові рішення, що визначають якість майбутньої бази даних закладаються на етапі розробки логічної моделі даних. "Хороші" моделі даних повинні відповідати певним критеріям:

- Адекватність бази даних предметної області
- Легкість розробки і супроводу бази даних
- Швидкість виконання операцій оновлення даних (вставка, оновлення, видалення)
- Швидкість виконання операцій вибірки даних

Перша нормальна форма (1НФ) - це звичайне відношення. Ставлення в 1НФ має такі властивості:

- У відношенні немає однакових кортежів.
- Кортежі не впорядковані.
- Атрибути не впорядковані.
- Всі значення атрибутів атомарний.

Відносини, що знаходяться в 1НФ є "поганими" в тому сенсі, що вони не задовольняють обраним критеріям - є велика кількість аномалій оновлення, для підтримки цілісності бази даних потрібна розробка складних тригерів.
Відношення *R* знаходиться під *другий нормальної формі (2НФ)* тоді і тільки тоді, коли відношення знаходиться в 1НФ і немає неключових атрибутів, залежних від частини складного ключа.

Відносини в 2НФ "краще", ніж в 1НФ, але ще недостатньо "хороші" - залишається частина аномалій оновлення, як і раніше потрібні тригери, що підтримують цілісність бази даних.
Відношення *R* знаходиться в *третій нормальній формі (3НФ)* тоді і тільки тоді, коли відношення знаходиться в 2НФ і все неключових атрибути взаємно незалежні.
Відносини в 3НФ є найбільш "хорошими" з точки зору обраних нами критеріїв

- усунуті аномалії оновлення, потрібні тільки стандартні тригери для підтримки посилальної цілісності.

Перехід від ненормалізованих відносин до відносин в 3НФ може бути виконаний за допомогою *алгоритму нормалізації.*
Алгоритм нормалізації полягає в послідовній декомпозиції відносин для усунення функціональних залежностей атрибутів від частини складного ключа
(Приведення до 2НФ) і усунення функціональних залежностей неключових атрибутів один від одного (приведення до 3НФ).

Питання для самоконтролю:

1. Дайте визначення поняттю "Ставлення в третій нормальній формі"?
2. Що ви розумієте під терміном "Друга нормальна форма"?
3. Наведіть приклад ставлення в третій нормальній формі.
4. Поясніть особливості поняття "Функціонально-залежні атрибути".
5. Назвіть основні етапи нормалізації відносини до третьої нормальної форми.

Лекція 8. Елементи мови SQL

1. Оператори SQL.
2. Оператори DDL (Data Definition Language) - оператори визначення об'єктів бази даних
3. Оператори DML (Data Manipulation Language) – оператори маніпулювання даними
4. Оператори захисту і управління даними
5. Приклади використання операторів маніпулювання даними

У даній лекції розглядаються елементи мови SQL (Structured Query Language). Поточна версія стандарту мови SQL прийнята в 1992 р (Офіційна назва стандарту - Міжнародний стандарт мови баз даних SQL (1992) (International Standart Database Language SQL), неофіційна назва - SQL / 92, або SQL-92, або SQL2).
Документ, що описує стандарт, містить понад 600 сторінок. Ми дамо тільки деякі поняття мови.
Мова SQL став фактично стандартною мовою доступу до баз даних. Всі СУБД, які претендують на назву "реляційні", реалізують той чи інший діалект SQL. Багато нереляційні системи також мають в даний час засоби доступу до реляційних даних. Метою стандартизації є переносимість додатків між різними СУБД.
Потрібно зауважити, що в даний час, жодна система не реалізує стандарт SQL в повному обсязі. Крім того, у всіх діалектах мови є можливості, які не є стандартними. Таким чином, можна сказати, що кожен діалект - це надмножество деякого підмножини стандарту SQL. це ускладнює переносимість додатків, розроблених для одних СУБД в інші СУБД.
Мова SQL оперує термінами, дещо відмінними від термінів реляційної теорії, наприклад, замість "відносин" використовуються "таблиці", замість "кортежів" - "рядка", замість "атрибутів" - "колонки" або "стовпці".
Стандарт мови SQL, хоча і заснований на реляційній теорії, але в багатьох місцях відходить він неї. Наприклад, ставлення в реляційної моделі даних не допускає наявності

однакових кортежів, а таблиці в термінології SQL можуть мати однакові рядки. Є й інші відмінності.
Мова SQL є реляційно повним. Це означає, що будь-який оператор реляційної алгебри може бути виражений відповідним оператором SQL.

1. Оператори SQL
Основу мови SQL складають оператори, умовно розбиті не кілька груп по виконуваних функцій.
Можна виділити наступні групи операторів (перераховані не всі оператори SQL):
2. Оператори DDL (Data Definition Language) – оператори визначення об'єктів бази даних

- CREATE SCHEMA - створити схему бази даних
- DROP SHEMA - видалити схему бази даних
- CREATE TABLE - створити таблицю
- ALTER TABLE - змінити таблицю
- DROP TABLE - видалити таблицю
- CREATE DOMAIN - створити домен
- ALTER DOMAIN - змінити домен
- DROP DOMAIN - видалити домен
- CREATE COLLATION - створити послідовність
- DROP COLLATION - видалити послідовність
- CREATE VIEW - створити уявлення
- DROP VIEW - видалити уявлення

3. Оператори DML (Data Manipulation Language) - оператори маніпулювання даними

- SELECT - відібрати рядки з таблиць
- INSERT - додати рядки в таблицю
- UPDATE - змінити рядки в таблиці
- DELETE - видалити рядки в таблиці
- COMMIT - зафіксувати внесені зміни
- ROLLBACK - відкотити внесені зміни

4. Оператори захисту і управління даними

- CREATE ASSERTION - створити обмеження
- DROP ASSERTION - видалити обмеження
- GRANT - надати привілеї користувачу або додатку на маніпулювання об'єктами
- REVOKE - скасувати привілеї користувача або додатки

Крім того, є групи операторів установки параметрів сеансу, отримання інформації про базу даних, оператори статичного SQL, оператори динамічного SQL.
Найбільш важливими для користувача є оператори маніпулювання даними (DML).
5. Приклади використання операторів маніпулювання даними
INSERT - вставка рядків у таблицю Приклад 10.1. Вставка одного рядка в таблицю:
INSERT INTO P (PNUM, PNAME) VALUES (4, "Іванов");
Приклад 10.2. Вставка в таблицю кількох рядків, обраних з іншої таблиці (в таблицю TMP_TABLE вставляються дані про постачальників з таблиці P, мають номери, великі 2):

```
INSERT INTO
TMP_TABLE (PNUM, PNAME) SELECT
PNUM, PNAME FROM P
WHERE P.PNUM> 2;
```

UPDATE - оновлення рядків в таблиці: Приклад 10.3. Оновлення декількох рядків в таблиці:

```
UPDATE P
SET PNAME = "Пушник" WHERE
P.PNUM = 1;
```

DELETE - видалення рядків в таблиці Приклад 10.4. Видалення кількох рядків у таблиці: DELETE FROM P WHERE P.PNUM = 1;
Приклад 10.5. Видалення всіх рядків в таблиці: DELETE FROM P;
Приклади використання оператора SELECT
Оператор SELECT є фактично найважливішим для користувача і найскладнішим оператором SQL. Він призначений для вибірки даних з таблиць, тобто він, власне, і реалізує одне з основних призначення бази даних - надавати інформацію користувачеві.
Оператор SELECT завжди виконується над деякими таблицями, що входять в базу даних.
Зауваження. Насправді в базах даних можуть бути не тільки постійно збережені таблиці, а також тимчасові таблиці і так звані уявлення. Уявлення - це просто зберігаються в базі дані SELECT-вирази. З точки зору користувачів уявлення - це таблиця, яка не зберігається постійно в базі даних, а "виникає" в момент звернення до неї. З точки зору оператора SELECT і постійно збережені таблиці, і тимчасові таблиці та подання виглядають абсолютно однаково. Звичайно, при реальному виконанні оператора SELECT системою враховуються відмінності між збереженими таблицями і уявленнями, але ці відмінності *приховані* від користувача.
Результатом виконання оператора SELECT завжди є таблиця. Таким чином, за результатами дій оператор SELECT схожий на оператори реляційної алгебри. Будь-оператор реляційної алгебри може бути виражений відповідним чином сформульованим оператором SELECT. Складність оператора SELECT визначається тим, що він містить в собі всі можливості реляційної алгебри, а також додаткові можливості, яких в реляційній алгебрі немає.
Відбір даних з однієї таблиці Приклад 10.6. Вибрати всі дані з таблиці постачальників (ключові слова *SELECT FROM):*

```
SELECT *
FROM P;
```

Зауваження. В результаті отримаємо нову таблицю, яка містить повну копію даних з вихідної таблиці P.
Приклад Вибрати всі рядки з таблиці постачальників, які задовольняють деякому умові (ключове слово *WHERE):*

```
SELECT *
FROM P
WHERE P.PNUM> 2;
```

Зауваження. Як умова в розділі WHERE можна використовувати складні логічні вирази, що використовують поля таблиць, константи, порівняння (>, <, = і т.д.), дужки, союзи AND і OR, заперечення NOT.
Приклад 10.8. Вибрати деякі колонки з вихідної таблиці (вказівка списку відбираються колонок):

```
SELECT P.NAME
FROM P;
```

Зауваження. В результаті отримаємо таблицю з одного колонкою, що містить всі найменування постачальників.
Зауваження. Якщо у вихідній таблиці були присутні кілька постачальників з різними номерами, але однаковими найменуваннями, то в результатірующего таблиці *будуть рядки з повтореннями* - дублікати рядків автоматично, не відкидаються.

Приклад 10.9. Вибрати деякі колонки з вихідної таблиці, видаливши з результату повторювані рядки (ключове слово *DISTINCT):*

```
SELECT DISTINCT P.NAME FROM P;
```

Зауваження. Використання ключового слова DISTINCT призводить до того, що в результатірующего таблиці будуть видалені всі повторювані рядки.

Приклад Використання скалярних виразів і перейменувань колонок в запитах (ключове слово *AS):*

```
SELECT
TOVAR.TNAME,
TOVAR.KOL,
TOVAR.PRICE,
 "=" AS EQU,
TOVAR.KOL * TOVAR.PRICE AS SUMMA FROM TOVAR;
```

В результаті отримаємо таблицю з колонками, яких не було у вихідній таблиці TOVAR:

TNAME	KOL	PRICE	EQU	SUMMA
Болт	10	100	=	1000
Гайка	20	200	=	4000
Винт	30	300	=	9000

Приклад Впорядкування результатів запиту (ключове слово *ORDER BY):*

```
SELECT PD.PNUM,
PD.DNUM,
PD.VOLUME
FROM PD
ORDER BY DNUM;
```

В результаті отримаємо наступну таблицю, впорядковану по полю DNUM:

PNUM	DNUM	VOLUME
1	1	100
2	1	150
3	1	1000
1	2	200
2	2	250
1	3	300

Приклад Впорядкування результатів запиту по декількох полях зі зростанням або убуванням (ключові слова *ASC, DESC):*

```
SELECT
PD.PNUM,
PD.DNUM,
PD.VOLUME FROM
PD ORDER BY DNUM
ASC, VOLUME DESC;
```

В результаті отримаємо таблицю, в якій рядки йдуть в порядку зростання значення поля DNUM, а рядки, з однаковим значенням DNUM йдуть в порядку убування значення поля VOLUME:

PNUM	DNUM	VOLUME
3	1	1000
2	1	150
1	1	100
2	2	250
1	2	200
1	3	300

Зауваження. Якщо явно не вказані ключові слова ASC або DESC, то за замовчуванням приймається впорядкування по зростанню (ASC).
Відбір даних з декількох таблиць
Приклад
Природне з'єднання таблиць (спосіб 1 - явне вказівку умов з'єднання):

```
SELECT
P.PNUM,
P.PNAME,
PD.DNUM,
PD.VOLUME
FROM P, PD
WHERE P.PNUM = PD.PNUM;
```

В результаті отримаємо нову таблицю, в якій рядки з даними про постачальників з'єднані з рядками з даними про постачання деталей:

PNUM	PNAME	DNUM	VOLUME
1	Иванов	1	100
1	Иванов	2	200
1	Иванов	3	300
2	Петров	1	150
2	Петров	2	250
3	Сидоров	1	1000

Зауваження. Сполучаються таблиці перераховані в розділі FROM оператора, умова з'єднання наведено в розділі WHERE. Розділ WHERE, крім умови з'єднання таблиць, може також містити і умови відбору рядків.
Приклад Природне з'єднання таблиць (спосіб 2 - ключові слова *JOIN USING):*

```
SELECT
P.PNUM,
P.PNAME,
PD.DNUM,
PD.VOLUME
FROM P JOIN PD USING PNUM;
```

Зауваження. Ключове слово USING дозволяє *явно вказати,* по яким з *загальних* колонок таблиць буде проводитися з'єднання.
Приклад Природне з'єднання таблиць (спосіб 3 - ключове слово *NATURAL JOIN):*

```
SELECT
P.PNUM,
P.PNAME,
PD.DNUM,
PD.VOLUME
FROM P NATURAL JOIN PD;
```

Зауваження. У розділі FROM не вказано, по яких полях здійснюється з'єднання. NATURAL JOIN автоматично з'єднує *по всіх однаковим полях* в таблицях.

Приклад Природне з'єднання трьох таблиць: SELECT

```
P.PNAME,
D.DNAME,
PD.VOLUME
FROM
P NATURAL JOIN PD NATURAL JOIN D;
```

В результаті отримаємо наступну таблицю:

PNAME	DNAME	VOLUME
Иванов	Болт	100
Иванов	Гайка	200
Иванов	Винт	300
Петров	Болт	150
Петров	Гайка	250
Сидоров	Болт	1000

Питання для самоконтролю:

1. Наведіть синтаксис операції SELECT
2. Перерахуйте оператори мови визначення даних.
3. Перерахуйте оператори мови манітпулірованія даними.

2. Плани семінарських занять, завдання для лабораторних робіт, самостійної роботи:

1. Загальні методичні вказівки

Аудиторні заняття з дисципліни «Основи організації бази даних» проводяться у формі лекційних занять та семінарських занять, на яких студенти повинні:

Вивчити: сучасні теорії організації баз даних, технології їх розробки (концепція та технологія баз даних; логічний рівень опису даних, ключі, зв'язки, схему даних; архітектуру баз даних; моделі даних і типи баз даних; реляційні бази даних та їх мови, мову запитів SQL; системи управління базами даних, її функції та характеристики; принципи побудови баз даних, принципи проведення аналізу предметної області..

Вміти:

для заданої предметної області, проектувати схему бази даних; працювати з базами даних засобами SQL; використовувати СУБД для роботи з БД.

На лекційних та семінарських заняттях з дисципліни використовуються методи навчання: пояснення теоретичного матеріалу, моделювання ситуацій з використанням мультимедійних технологій, варіативне виконання індивідуальних завдань з використанням комп'ютерної техніки.

Вивчення рекомендованої літератури є основним джерелом знань із курсу. При проведенні аудиторних занять, передбачається підготовка рефератів; здійснення контролю рівня засвоєння матеріалу за допомогою поточного тестування або контрольних робіт за темами та підсумкового тестування.

Основною формою вивчення більшості питань тем курсу є самостійна підготовка студентів до семінарських занять. Це зумовлено:

- значним обсягом навчального матеріалу;
- обмеженим обсягом аудиторних занять.

Результати участі кожного студента на аудиторних заняттях, його самостійної роботи і засвоєння навчального матеріалу оцінюються викладачем.

Критерії оцінювання роботи студентів на аудиторних заняттях:

- оцінка «відмінно» - студент повно і всебічно розкриває питання теми, які винесено на самостійне опрацювання, вільно оперує поняттями і термінологією, демонструє глибокі знання джерел, має власну точку зору стосовно відповідної теми і може аргументовано її доводити;
- оцінка «добре» - загалом рівень знань студентів (слухачів) відповідає викладеному вище, але спостерігаються деякі недоліки при виконанні завдань;
- оцінка «задовільно» - студент розкрив питання, винесені на самостійне опрацювання, аудиторне заняття в загальних рисах, розуміє їх суть, намагається робити висновки, але при цьому припускається грубих помилок, матеріал викладає нелогічно;

- оцінка «незадовільно» - студент не в змозі дати відповідь на поставлене запитання або відповідь неправильна, не розуміє суті питання, не може зробити висновків.

По закінченні вивчення курсу проводиться підсумковий контроль. Студент, який пропустив семінарське заняття (незалежно від причини пропуску), не з'явився на кафедру для індивідуального відпрацювання матеріалу у встановлені дні і години, до підсумкового контролю не допускається.

2. Структура навчальної дисципліни

Назви розділів і тем	Кількість годин											
	денна форма						заочна форма					
	усього	у тому числі					усього	у тому числі				
		л	п	лаб.	інд.	с. р.		л	п	лаб.	інд.	с. р.
1	2	3	4	5	6	7	8	9	10	11	12	13
Розділ 1. Фундаментальні поняття баз даних і систем управління базами даних												
Тема 1.	9	2	4			3						
Тема 2.	11	2	4		2	3						
Тема 3.	11	2	4		2	3						
Тема 4.	11	2	4		2	3						
Разом за розділом 1	42	8	16		6	12						
Розділ 2. Принципи проектування баз даних												
Тема 1.	12	2	4		3	3						
Тема 2.	14	2	4		2	6						
Тема 3.	11	2	4		2	3						
Тема 4.	11	2	4		2	3						
Разом за розділом 2	48	8	16		9	15						
Усього годин	90	16	32		15	27						

Робота 1,2 Створення нової бази даних та таблиць MS Access.

Мета: опанувати технологію зберігання інформації в інформаційних системах за допомогою реляційних баз даних.

Технічне обладнання

Хід роботи

1. Під час запуску Microsoft Access з'являється діалогове вікно (Рисунок 1), в якому пропонується створити нову базу даних або відкрити вже існуючу. При появі цього діалогового вікна оберіть пункт *«Новая база данних»* та натисніть кнопку ОК. Якщо база даних вже була відкрита або закрите вікно запуску, натисніть кнопку *«Создать новую базу данных»* на панелі інструментів та двічі натисніть кнопку миші, встановив курсор на позначку нової бази даних. Вкажіть ім'я та каталог нової бази даних і натисніть кнопку ОК.

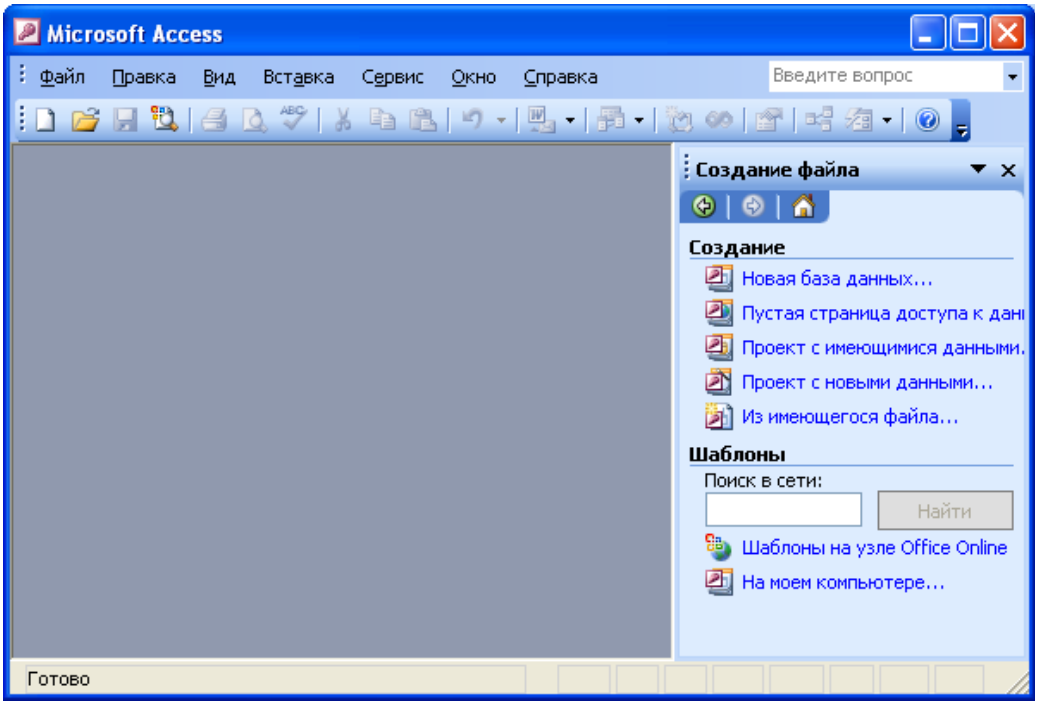
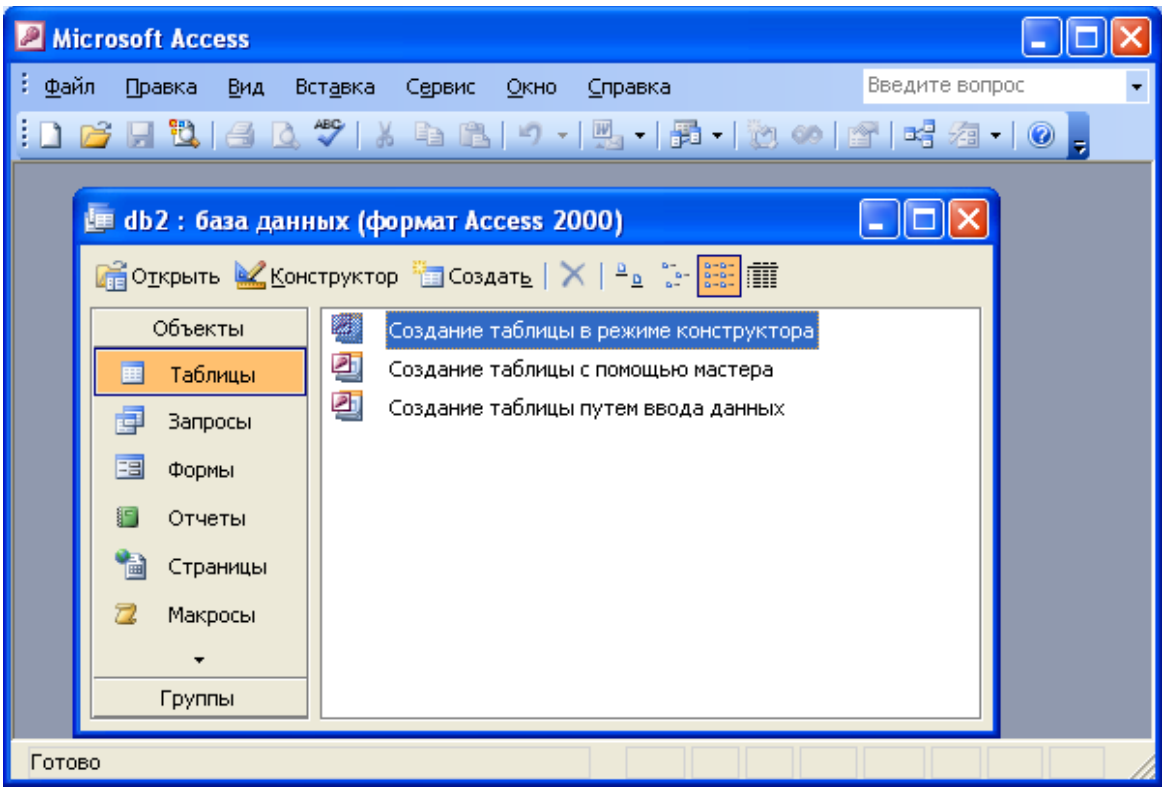


Рисунок 1 – Діалогове вікно створення нової бази даних.

Після створення порожньої бази даних необхідно самостійно створити об'єкти цієї бази. В базі даних MS Access існує шість типів об'єктів: таблиці, запити, форми, звіти, макроси та модулі (Рисунок 2).

2. Робота с базою даних починається з створення таблиць. Для створення таблиці в режимі конструктора необхідно перейти головне вікно бази даних (Рисунок 2) та вибрати об'єкт *«Таблица»* та двічі натиснути кнопку миші на елементі *«Создание таблицы в режиме конструктора»*.



2. Створення таблиці у режимі конструктора.

3. Наступним кроком є визначення структури таблиці. При визначенні структури таблиці в режимі конструктора необхідно виконати наступні кроки (Рис.3).

— Оберіть стовпець *«Имя поля»* та введіть ім’я поля, відповідно до погодження щодо імен об’єктів Microsoft Access.

— В стовпці *«Тип данных»* виберіть необхідний тип даних у списку.

— В стовпці *«Описание»* введіть опис (коментарій), яке розташовується у цьому полі. Текст опису буде з’являтися в рядку стану під час додавання даних у поле, а також буде включеним в опис об’єкта таблиці. Додавання опису є необов’язковим.

— При необхідності додайте значення властивостей поля в бланку властивостей у нижній частині вікна.

4. Визначте ключові поля таблиці. Для цього необхідно виділити поля, які мають бути ключем та нажати піктограму з зображенням ключа на панелі інструментів. Рекомендується визначати ключові поля, але це не

обов’язково. Якщо вони не були визначені, то під час збереження таблиці видається питання, чи треба їх створити.

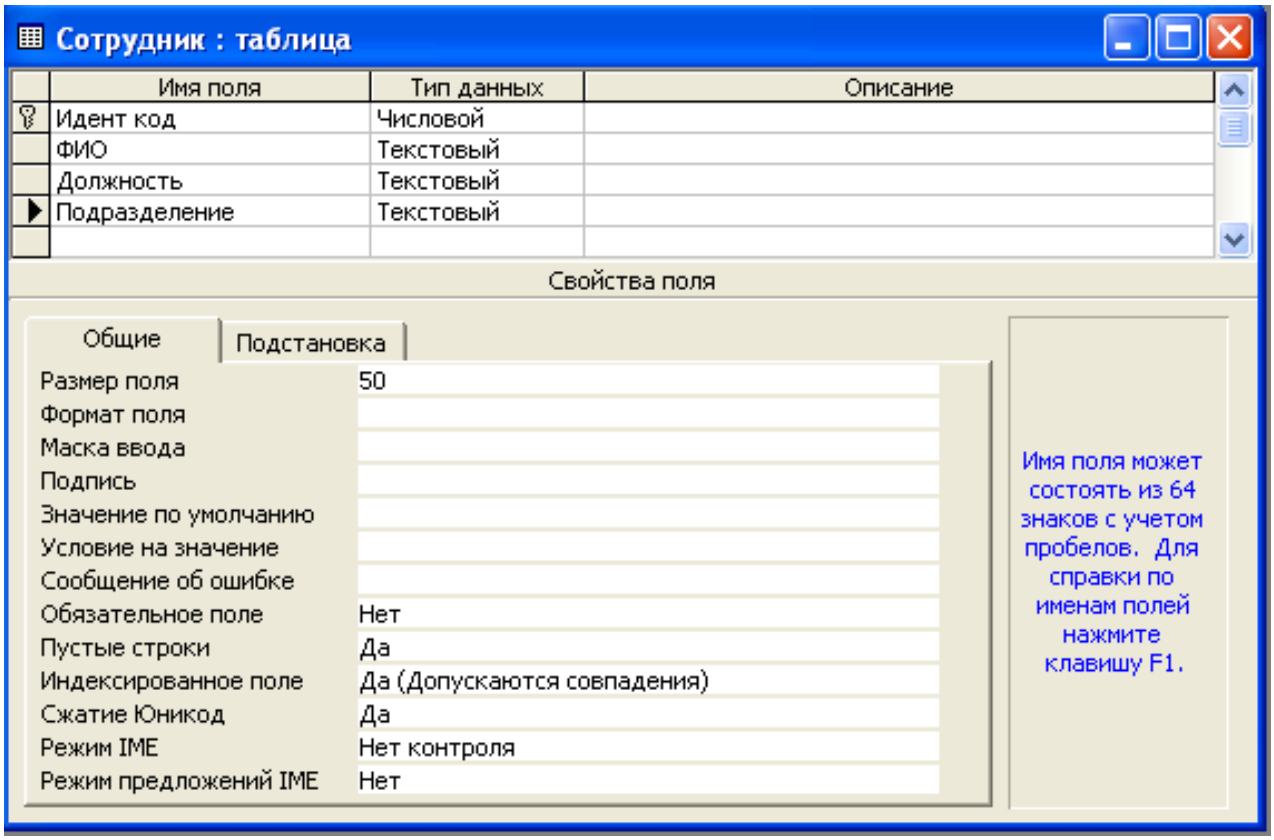


Рисунок 3. Визначення структури таблиці в режимі конструктора

5. Збереження таблиці. Для збереження таблиці натисніть кнопку *«Сохранить»* на панелі інструментів (ця кнопка має вигляд дискети), а далі введіть ім’я таблиці у відповідності з погодженням щодо імен об’єктів Microsoft Access.

Під час роботи з об’єктом *„Таблица”* можливо користуватися двома режимами: режимом конструктора та режимом таблиць (Рисунок 4). Режим конструктора використовується для створення та внесення змін до структури таблиці, а режим таблиці для перегляду, додавання, вилучення та редагування даних в таблиці.

Переключення між режимами за допомогою піктограми *Вид».* Якщо таблиця відкрита в режимі таблиці, то на екрані відображена лише кнопка переключення в режим конструктора та навпаки (Рисунок 4).

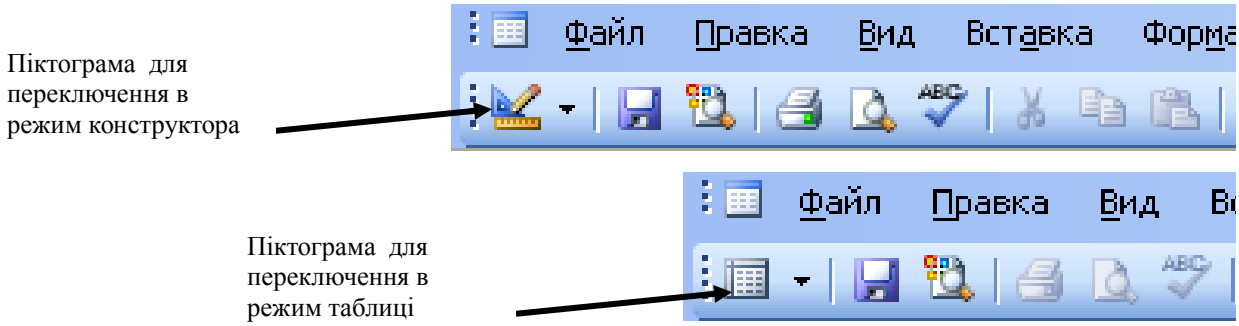


Рисунок 4 Переключення між режимами за допомогою піктограми *Вид»*

Після того, як таблиця створена та її структура описана в режимі конструктора, таблицю можна заповнювати даними. Для цього необхідно відкрити таблицю в режимі таблиці (Рисунок 5). У цьому режимі також можливо сортирувати та фільтрувати записи.

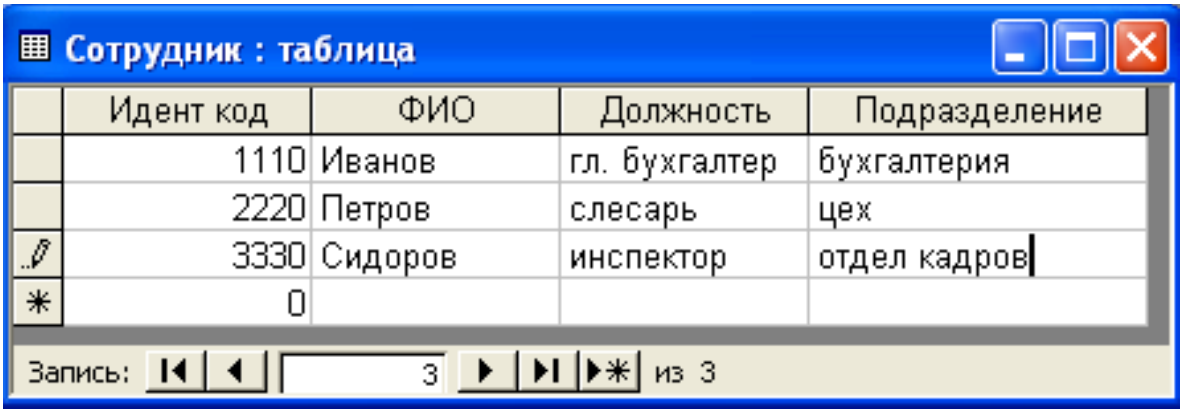


Рисунок 5. Заповнення таблиці даними

Завдання 1

Описати структуру таблиць для реляційної бази даних за допомогою якої автоматизується процес управління тогами набором фінансових активів - цінних паперів (при відносних обмеженнях об’ємів інформації). В торговій системі існує певний набір цінних паперів (акцій), кожна характеризується

найменуванням, номінальною ціною, сумарним об’ємом пакету (скільки всього одиниць даного паперу було емітовано), датою емісії. Одночасно на ринку діють агенти, які продають або покупають папери. Кожний агент характеризується найменуванням та кількістю грошових коштів. Таким чином мають бути чотири масиви інформації: дані по цінним паперам, дані по агентам, дані по портфелям (належність паперів агентам), дані по заявкам на покупку або продаж паперів.

Завдання 2

Створити структуру таблиць для фрагменту реляційної бази даних за допомогою якої автоматизується облік основних засобів. Основні об’єкти предметної області та їх характеристики: група ОЗ (код групи ОЗ, найменування групи, річна норма амортизації,); МВО - матеріально-відповідальна особа (табельний номер, ПІПБ); інвентарний об’єкт ОЗ(інвентарний номер ОЗ, код групи ОЗ, код підрозділу, початкова вартість, начальний знос); амортизація ОЗ (інвентарний номер ОЗ, дата амортизації, сума амортизації); інвентарна карточка ОЗ (номер карточки ОЗ, інвентарний номер ОЗ, табельний номер, код підрозділу, дата відкриття карточки, дата закриття карточки).

Контрольні запитання

1. Що таке база даних та що таке СУБД? В чому різниця?
2. До якого класу програм належить MS Access?
3. До виду забезпечення інформаційних систем належить MS Access?
4. Дайте визначення наступним поняттям: об’єкт, атрибут.
5. Назвіть типи даних, які використовуються в MS Access.

Робота 3, 4 Пов’язування таблиць. Створення схеми даних.

Мета: опанувати технологію опису логічних зв’язків між таблицями MS Access

Технічне обладнання

Хід роботи

Якщо дані таблиць бази даних логічно пов’язані, то ці зв’язки необхідно вказати. Для пов’язання таблиць необхідно їх проіндексувати за тими полями, по яким створюється зв’язок. Для цього необхідно в режимі конструктора таблиці присвоїти властивості «Індексоване поле» для потрібного поля таблиці значення *«Так…»*. При цьому необхідно також вказати чи припускаються збіги значень для даного поля (чи можливі в таблиці декілька записів з однаковим значенням індексуємого поля). Для створення зв’язку необхідно, щоб хоч для одного з пов’язаних полів збіги не припускалися. Поля таблиць, які пов’язуються повинні бути погодженими за типом та розміром.

Схема даних відображає таблиці бази даних, їх поля, а також зв’язки між таблицями в графічному вигляді. Для створення схеми даних необхідно виконати наступні.

1. Зберегти та закрити всі створені таблиці. Залишити тільки головне вікно бази даних.

2. При відкритому вікні бази даних оберіть пункти меню *«Сервис»* → *«Схема данних».*

3. У діалоговому вікні, що відкриється, необхідно послідовно обрати таблиці, які поміщуються до схеми даних і для кожної таблиці натиснути кнопку *«Добавить».*

4. Після додавання таблиць необхідно закрити діалогове вікно - кнопка *«Закрыть».*

5. Відкриється вікно схеми даних, у якому у вигляді прямокутників будуть відображені всі відібрані таблиці.

6. Встановіть зв’язки між таблицями, для чого необхідно перетягнути мишкою обране поле з однієї таблиці в іншу та відпустити кнопку над полем, з яким встановлюється зв’язок.

У діалоговому вікні *„Изменение связей”*, яке з’явиться можна вказати необхідність підтримання цілісності даних та каскадного оновлення/вилучення записів (Рисунок 6) .

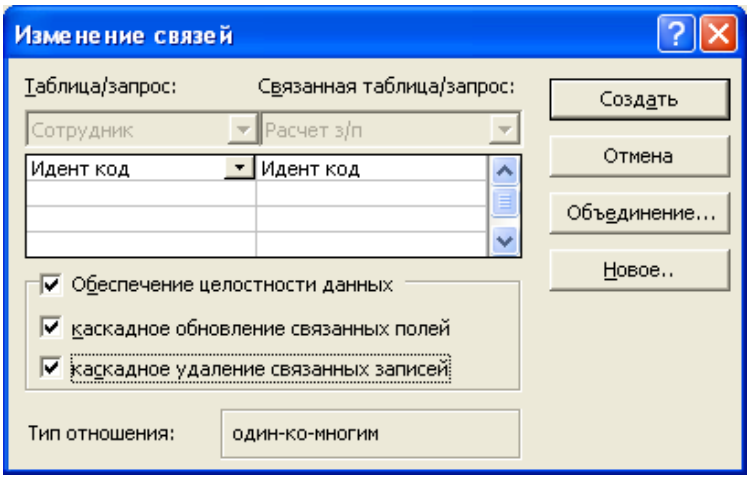


Рисунок 6 Діалогове віно *„Изменение связей”*

7. Закрити діалогове вікно за допомогою кнопки *«Создать»*. Отримана схема має вигляд вказаний на мал.6. Вид зв’язку (1→∞) можна побачити на Рисунку 7

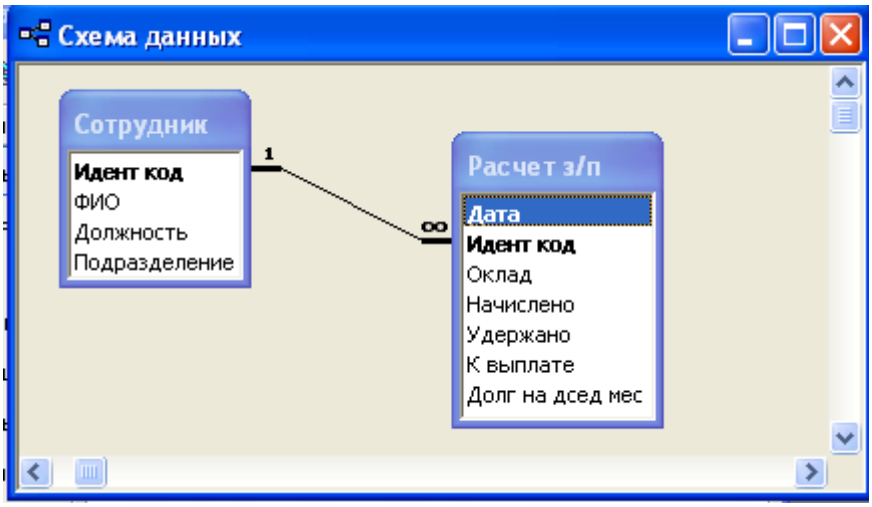


Рисунок 7. Схема даних

Завдання 1

Пов’язати таблиці, які були створені для автоматизації ринка торгівлі цінними паперами (Робота 1, Завдання 1). Пояснити зв’язки.

Завдання 2

Пов’язати таблиці, які були створені для автоматизації обліку основних засобів (Робота 1, Завдання 2). Пояснити зв’язки.

Контрольні запитання

1. Яки типи зв’язку Ви знаєте?
2. Що таке „каскадне оновлення даних”, „каскадне видалення даних”?
3. Дайте визначення поняттю ключ.
4. Що таке модель даних? Які моделі ви знаєте?
5. Назвіть основні властивості реляційної моделі даних.
6. Що таке нормальні форми?
7. Для чого служить схема даних MS Access?

Робота 5,6 Створення запиту на вибірку в режимі конструктора.

Мета: опанувати технологію

Технічне обладнання

Хід роботи

Під запитом в MS Access розуміють як команду на вибір, просмотр, зміну, створення, або видалення даних. Також запити використовуються для аналізу даних.

1. У вікні бази даних перейдіть до вкладки *«Запити»* та натисніть кнопку *«Створити».*

2. У діалоговому вікні «*Новий запит*» оберіть команду «*Конструктор*» та натисніть кнопку ОК.

3. У діалоговому вікні *«Додавання таблиці»* перейдіть до вкладки, яка містить об’єкти, які включають потрібні дані.

4. Для додавання об’єктів в запит двічі натисніть кнопку миші на імені кожного об’єкта, а потім натисніть кнопку *«Закрити»*.

5. Якщо запит містить декілька таблиць або запитів, переконайтесь, що між собою їх з’єднує лінія. Для Microsoft Access це свідчить про те, що дані пов’язані. Якщо зв’язку немає, то створіть його за допомогою діалогового вікна *«Схема даних»*. Якщо таблиці або запити пов’язані, то тип з’єднання можна змінити, маючи тим самим вплив на обрання записів у запиті.

6. Додайте поля до запиту, перетягуючи їх імена за допомогою миші із списку полів в бланк запиту. Зразок запиту в режимі конструктора для розібраних раніше таблиць можна побачити на рис.11. В цьому запиті потребується відобразити дані про робітників, в яких значення суми в полі „К выплате” більше або дорівнює 1000 грн. Записи в запиті необхідно розсортирувати по зростанню поля *«Прізвище»,* тобто по алфавіту. Тому поле „К выплате” містить вираз „>=100”, а також вказуємо необхідний порядок сортування даних.

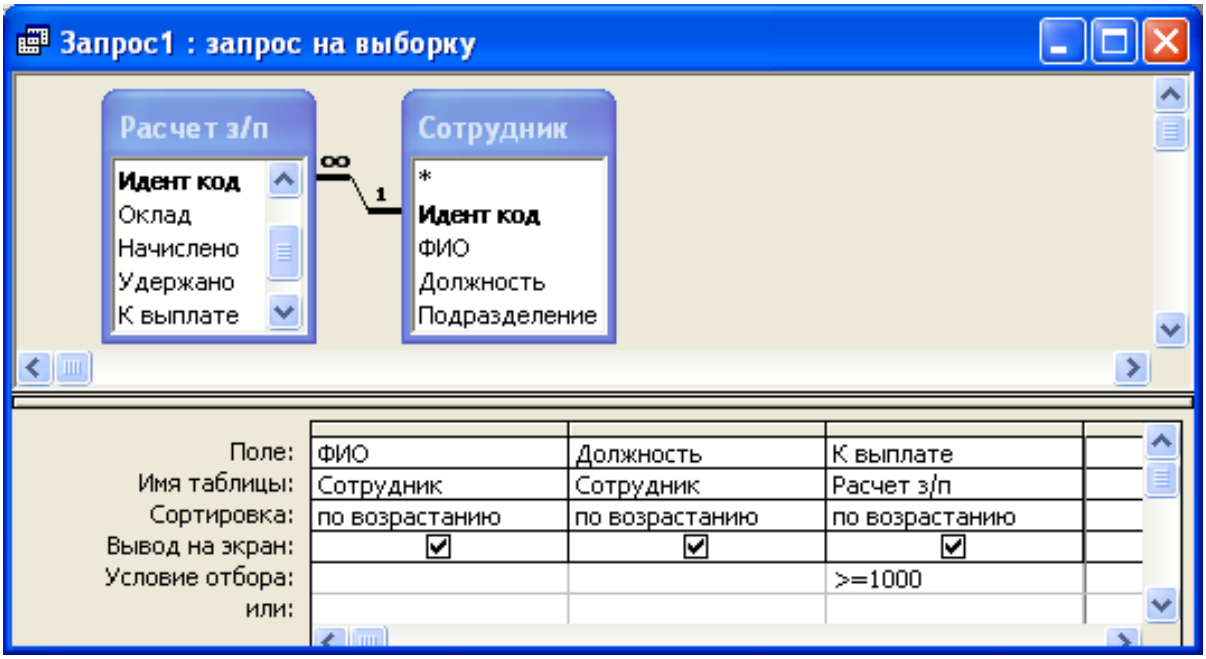


Рисунок 11 - запит в режимі конструктора

7. Для збереження запиту натисніть кнопку *«Зберегти»* на панелі інструментів. Введіть ім'я, що відповідає погодженням щодо імен об'єктів Microsoft Access, та натисніть кнопку ОК.

8. Для перегляду результатів запиту натисніть кнопку *«Вид»* на панелі інструментів. Результат виконання запиту можна побачити на мал.12.

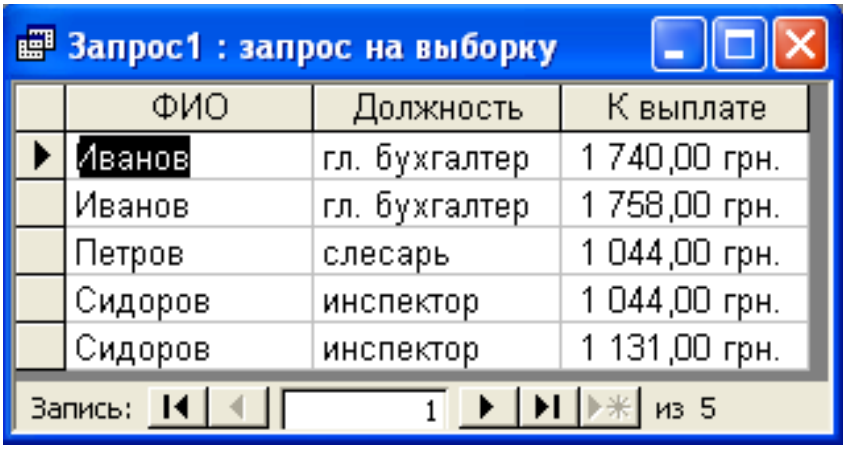


Рисунок 12 - запит в режимі таблиці

Розглянемо таблицю„Расчет з/п". На основі набору даних цієї таблиці бухгалтер може сформувати необхідні документи (відомість по заробітній платі, розрахунковий лист для кожного працівника та ін.). Доцільно буде сформувати запит, в якому дані таблиці „Расчет з/п" будуть доповнені даними з таблиці „Сотрудник", а також зробити необхідні розрахунки в полях „Удержано" і „К виплате".

При створенні структури запиту додаємо поле „ФИО" і вказуємо сортируваня по зростанню (тобто по алфавіту). В полі „Удержано" треба написати вираз „Удержано: [Начислено]*0,13",а в полі „К виплате" – „К выплате: [Начислено]-[Удержано]".

Вигляд запиту в режимі конструктора наведено на рисунку 13

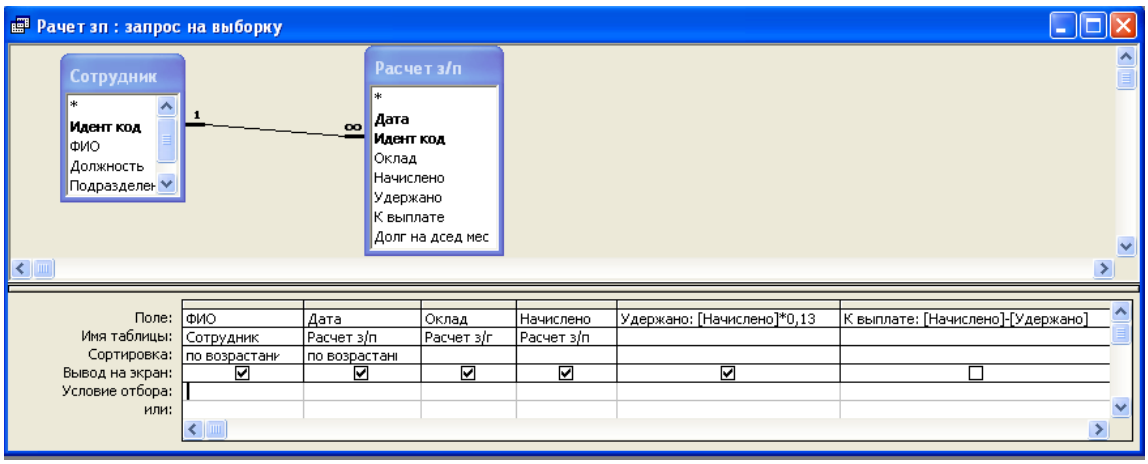


Рисунок 13 - Расчет з/п - запит в режимі конструктора

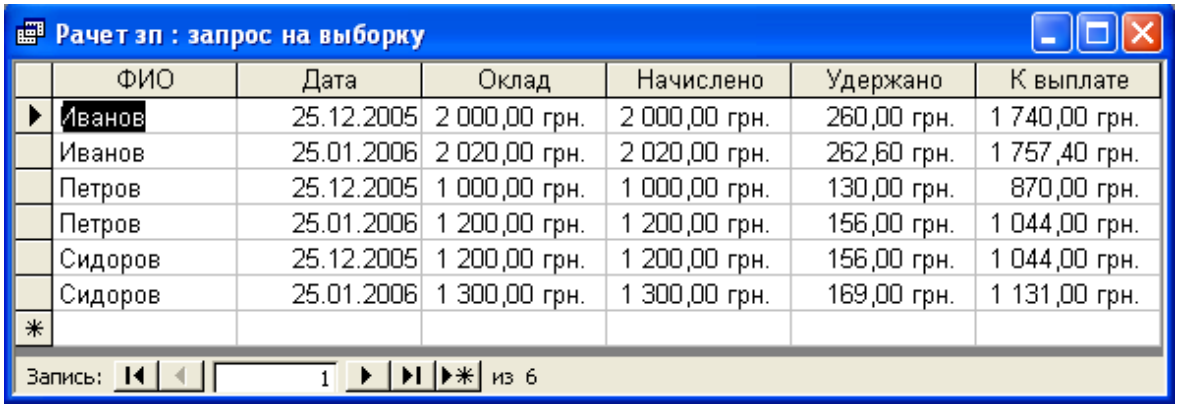


Рисунок 14 - Расчет з/п - запит в режимі таблиці

Завдання 1

1) Створити запит для визначення сумарного попиту, та пропозиції на цінні папери, які циркулюють на ринку.
2) Слідуючи логіці задачі автоматизації торгівлі цінними паперами, самостійно створити запит для відбору даних.

Завдання 2

1) Створити запит для відбору даних з БД обліку основних засобів за критерієм:
   - річна норма амортизації більше (або менше) певної суми;

Контрольні запитання

1) Що таке запит в MS Access.

2) Що таке процес „нормалізації” таблиць, його призначення? Сформулюйте I, II, III нормальні форми.
3) Які класи СУБД ви знаєте, чим вони відрізняються? Наведіть приклади.
4) Де застосовуються СУБД.

Робота 7, 8 Створення екранних форм для роботи з даними за допомогою майстра форм.

Мета: опанувати технологію розробки елементів організації інтерфейсу ввода/вивода даних засобами MS Access.

Технічне обладнання

Хід роботи

Введення даних безпосередньо в таблицю (робота 1)має ряд недоліків:

- структура таблиці повинна створюватися на базі логіки задач, яка може відрізнятись від логіки накопичення та вводу даних.
- якість автоматизованої системи визначається множиною показників одним з них це організація системи ввода/вивода, яка повинна бути максимально наближена до традиційних форм подання інформації на немашинних носіях. Такі форми, як правило, роблять програмне забезпечення більш привабливим для кінцевого користувача, та зменшують період його адаптації дот системи та дозволяють швидко зосередитися на вирішенні основних професійних задач;
- в автоматизована інформаційна система повинна забезпечувати розподілення прав доступу до даних в залежності від функцій, які виконують користувачі.

Форми створюються для вводу та редагування даних таблиць, перегляду результатів запитів, створення меню для користувачів, діаграм.

1. У головному вікні бази даних оберіть об’єкт *«Форми»* та двічі натиснути кнопку миші на елементі *«Создание формы с помощью мастера форм»*.

2. Оберіть ім’я таблиці або запиту, які містять дані, на основі яких буде створена форма та натисніть кнопку ОК.

3. Оберіть поля таблиці або запиту, які будуть відображені на формі, та натисніть кнопку *„Далее”* (Рисунок 8).

4. Якщо на кроці 3 були обрані *«Майстер форм», «Діаграма»* або *«Вільна таблиця»*, то натисніть кнопку ОК та додержуйтесь інструкцій, які з’являються в діалогових вікнах відповідного майстра. При обиранні елементів *«Автоформа: в стовпець», «Автоформа: стрічкова»* або *«Автоформа: таблична»* форма створюється автоматично.

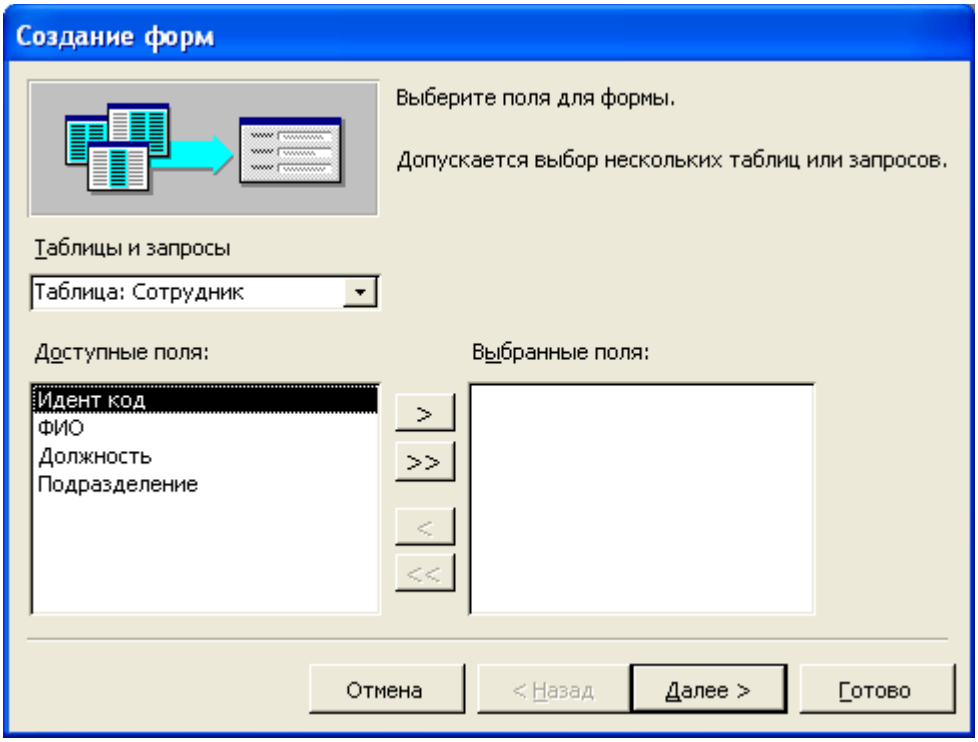


Рисунок 8. Створення форми за допомогою майстра форм

5. Якщо на кроці 3 були обрані *«Майстер форм», «Діаграма»* або *«Вільна таблиця»*, то натисніть кнопку ОК та додержуйтесь інструкцій, які з’являються в діалогових вікнах відповідного майстра. При обиранні

елементів *«Автоформа: в стовпець», «Автоформа: стрічкова»* або *«Автоформа: таблична»* форма створюється автоматично.

6. Для створення форми неавтоматичним засобом використовується режим *«Конструктор форм»*. При цьому користувач самостійно розміщує та форматує елементи даних (поля) на формі.

Примітка. При обрані однієї з автоформ використовується автоформат, який був останній раз встановлений в майстрі форм або за допомогою команди *«Автоформат»* в меню *«Формат»* у режимі конструктора.

Зразки форм, створених за допомогою *«Майстра форм»* приведені на мал.9-10.

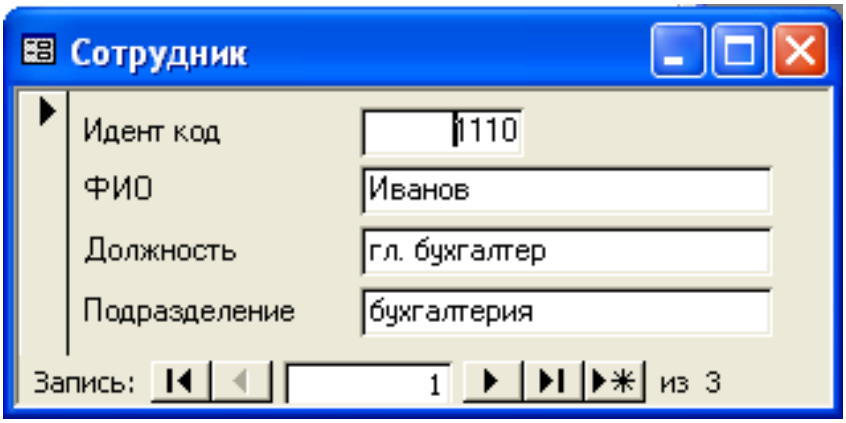


Рисунок 9.- Форма для таблиці „Співробітник” в один стовпець

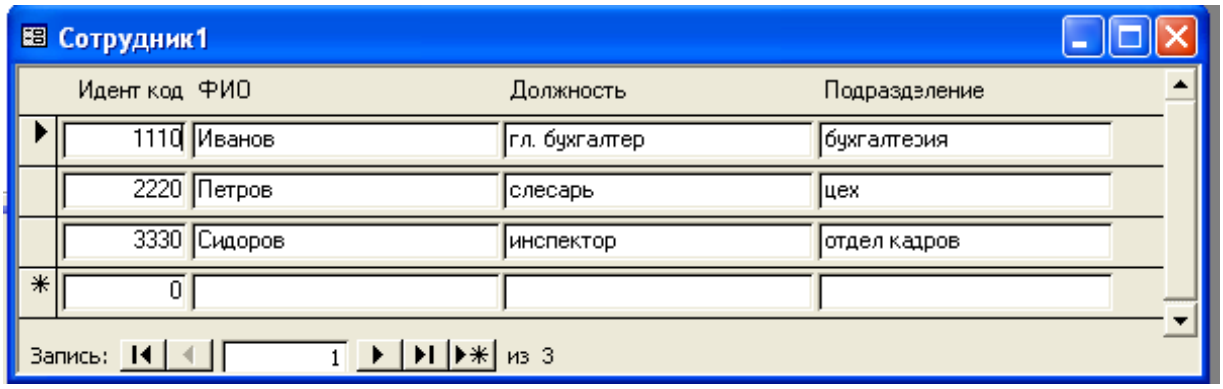


Рисунок 10.- Форма для таблиці „Співробітник” в табличній формі

Завдання 1

1) Для задачі автоматизації процесу управління торгівлею цінними паперами створити форми для роботи з таблицями „Папери” та „Агенти”.
2) Для задачі автоматизації процесу управління торгівлею цінними паперами створити форму „Портфелі” , яка буде створена на базі таблиць „Папери”, „Агенти”, „Портфелі” та містити додаткову інформацію про номінальну вартість пакету, яким володіє кожний агент в полі „Номінальна вартість пакету”.
3) Створити форму „Замовлення”,

Завдання 2

1) Для задачі автоматизації обліку основних засобів створити форми для роботи з таблицями „Інвентарна карточка ОЗ”, „Матеріально-відповідальна особа”, „Інвентарний об’єкт ОЗ”, „Група ОЗ”.
2) Слідуючи логіці задачі автоматизації обліку основних засобів створити форму на базі декількох або запитів.
3) Відкрити форму „Інвентарна карточка ОЗ” в режимі „Конструктор” визначити властивості форми. Змінити колір форми.

Контрольні запитання

1. Назвіть основні етапи створення бази даних засобами MS Access.
2. Основне призначення форм в автоматизованих системах.
3. Які способи створення форм ви знаєте?
4. Які джерела даних можуть бути використані для створення форм в MS Access?
5. Які властивості форми задаються користувачем при створенні форми за допомогою майстра?
6. Які властивості має форма в режимі „Конструктор” ?

Робота 9, 10 Створення звітів з допомогою майстра.

Мета: опанувати технологію створення

Технічне обладнання

Хід роботи.

Звіти це традиційна форма представлення інформації для управляння. Звіти виводяться на екран монітора, принтер, або в файл та створюються на базі таблиць і запитів. Технологія створення звітів MS Access майже ідентична технології створення форм.

Зарплата 2005-2006 год

Сотрудник_Идент код ФИО	Дата з/п	з/п_Идент код	Оклад	Начислено	Удержано	К выплате	Долг на дсед мес
1110 Иванов							
	25.12.2005	1110	2 000,00 грн.	2 000,00 грн.	260,00 грн.	1 740,00 грн.	0,00 грн.
	25.01.2006	1110	2 020,00 грн.	2 020,00 грн.	262,60 грн.	1 758,00 грн.	0,00 грн.
2220 Петров							
	25.12.2005	2220	1 000,00 грн.	1 000,00 грн.	130,00 грн.	870,00 грн.	0,00 грн.
	25.01.2006	2220	1 200,00 грн.	1 200,00 грн.	156,00 грн.	1 044,00 грн.	0,00 грн.
3330 Сидоров							
	25.12.2005	3330	1 200,00 грн.	1 200,00 грн.	156,00 грн.	1 044,00 грн.	0,00 грн.
	25.01.2006	3330	1 300,00 грн.	1 300,00 грн.	169,00 грн.	1 131,00 грн.	0,00 грн.

Рисунок 10 - Звіт

Завдання 1

Створити розрахунковий лист для кожного працівника за звітний період

3. Питання, задачі, завдання або кейси для поточного та підсумкового контролю знань і вмінь здобувачів вищої освіти, для контрольних робіт, передбачених навчальним планом, після атестаційного моніторингу набутих знань і вмінь з навчальної дисципліни:

Тема контрольної роботи "Розробка схеми даних завдання"

Найменування завдання".

В якості індивідуальних завдань передбачено виконання студентами контрольної роботи, що містить теоретичну та практичну частини. Практичне завдання виконується засобами MS Access. Для виконання теоретичної частини студент обирає питання з наступного переліку:

1. Тенденції та перспективи розвитку інформаційних систем, систем управління базами даних та базами знань;
2. Технології збереження, пошуку та обробки інформації;
3. Теоретичні основи побудови та функціонування баз даних і баз знань, характеристики сучасних СУБД, сучасні технології організації БД;
4. Правила розробки структури баз даних та створення прикладного програмного забезпечення з використанням систем управління базами даних;
5. Принципи побудови та технологію проектування баз даних і баз знань;
6. Основні поняття реляційної моделі даних;
7. Основи мови побудови запитів мовою SQL;
8. Засоби побудови баз даних за допомогою MS Access;
9. Використання інформаційних систем у різних предметних сферах;
10. Проектування інформаційних систем, баз даних та баз знань;
11. Створення програмного забезпечення для доступу до баз даних (засобами MS Access);
12. Аналіз даних засобами сучасних систем управління базами даних.
13. Реляційна база даних.
14. Що таке відношення? Що таке ключ?
15. Для чого використовуються зв’язки між таблицями? Як здійснюється зв’язування таблиць?
16. Для чого використовується майстер підстановки?
17. Що визначає властивість поля “Присоединенный столбец”?

18. Що визначає властивість поля “Число столбцов”? Що визначає властивість поля “Ширина столбцов”?

19. Як визначити для підстановки поле зі списком з виведенням на екран двох стовбців?

20. Як приховати виведення стовпця ключа на екран при використанні підстановки?

21. Як визначити зв’язок між таблицями у MS Access?

22. Що таке каскадне відновлення у таблицях MS Access?

23. Що таке каскадне видалення даних у таблицях MS Access?

24. Дайте визначення першої, другої, третьої та четвертої нормальних форм.

25. Опишіть процедуру створення таблиці у MS Access.

26. Назвіть типи даних, які можна використовувати при визначенні таблиць у MS Access.

27. Як здійснити перевірку правильність значень, які користувач вводить у таблицю?

28. Як визначити повідомлення при введенні неправильного значення, яке користувач вводить у таблицю?

29. Що таке індекс? Як визначити індексоване поле?

30. Що таке запит? Які види запитів існують в MS Access?

31. Назвіть способи об’єднання таблиць у запитах.

32. Як створити запит на основі фільтра?

33. Що таке обчислюване поле? Правила побудови виразів у MS Access.

34. Побудова виразів засобами «Построителя выражений».

35. Використання вбудованих функцій у майстрі побудови виразів.

36. Як формуються прості умови відбору записів?

37. Як формуються складені умови відбору записів?

38. Як знайти унікальні значення деякого поля таблиці?

39. Які агрегатні функції можна використовувати в запитах?

40. Наведіть приклади запитів з використанням логічного оператора AND, OR.

41. Наведіть приклади запитів з використанням оператора BETWEEN.

42. Наведіть приклади запитів з використанням логічного оператора IN.

43. Наведіть приклади запитів з використанням логічного оператора LIKE.

44. Як в запитах здійснюється групування даних?

45. Опишіть створення перехресного запиту за допомогою майстра.

Перед виконанням практичного завдання конрольної роботи контрольної роботи студент повинен обрати завдання з обліку відомої йому предметної області. Предметна область вважається відомою, якщо студент знайомий з бізнес процесом і потоками даних завдання.

Приклади функціональних завдань:

"Облік пацієнтів" ІС поліклініки;

"Облік договорів з постачальниками матеріалів" підсистеми "Відділ постачання;

"Облік роботи класного керівника" ІС "Школа" і т.п.

Зміст контрольної роботи:

1.Описание об'єкта автоматизації

2.Концептуальное проектування

3. Логічне проектування

У першому розділі наводиться опис об'єкта автоматизації як показано в прикладі нижче.

У другому розділі виділяються типи сутностей, зв'язку, описуються атрибути сутностей, домени атрибутів. Всі ці відомості наводяться в вигляді таблиць, як показано в прикладі нижче. Закінчується розділ побудовою концептуальної схеми даних завдання у вигляді ER-діаграми.

У третьому розділі необхідно провести перетворення розробленої

концептуальної моделі даних в реляційну схему даних, усунути з цим багато-до-багатьох і застосувати правила нормалізації. Виконані розробки описуються в текстовому вигляді і наводиться логічна схема даних завдання у вигляді удосконаленої ER-діаграми.

Приклад опису об'єкта автоматизації

Компанія під назвою DreamHome займається здачею в оренду об'єктів нерухомості за дорученням їх владельцев.Компанія пропонує повний комплекс послуг власникам, які бажають здати в оренду свою мебльовану нерухомість. Пропоновані компанією DreamHome послуги включають рекламу нерухомості в пресі, опитування передбачуваних орендарів, організацію перегляду здаються в оренду об'єктів потенційним орендарям, а також складання договорів на оренду. Після здачі нерухомості в оренду на компанію DreamHome покладається відповідальність за неї, тобто співробітники DreamHome повинні регулярно перевіряти поточний стан об'єктів.

Компанія DreamHome має кілька відділень, розташованих по всій території країни. Кожне відділення має унікальний номер і адресу, номер телефону та факсу. Кожне відділення имее свій власний штат співробітників.

У кожному відділенні компанії DreamHome.

Відомості про типи сутностей

Имя сущности	Опис	Псевдоніми	Особливості використання
Branch (відділення)	Місце роботи	Офіс	Одне або більше відділень компанії розміщені в різних містах
Staff (работник)	Загальний термін, що описує весь персонал, що працює в компанії		Кожен із співробітників працює в одному з відділень в компанії

Відомості про типи зв'язків

Тип сутності	Тип зв'язку	Тип сутності	Кардинальність
Branch (отделение)	Has (має)	Staff	1:M
Staff (працівник)	Managers (відполвідає за)	Property_for_Rent	1:M

Відомості про атрибути

Тип Сутності	Атрибут	Опис	Тип даних, довжина	Обмеження	Значення за замовчуванням	Допустимість Null
Branch	Branch_No	Унікальний ідентифікатор офісу відділення	Символьний, до 3 символів	Первинний ключ		Ні
	Street	Вулиця в адресі відділення	Символьний, до 25 символів			Ні
	Area	Район в адресі відділення	Символьний, до 15 символів			Ні
	City	Город в адресі відділення	Символьний, до 15 символів			Ні
	Postcode	Поштовий код в адресі відділення	Символьний, до 8 символів			Так
	Tel_No	Номер телефону відділення	Символьний фіксований 13 символів			Ні
	Fax_No	Номер факсу відділення	Символьний фіксований, 13 символів			Ні

Відомості про домени атрибутів

Ім'я домену	Характеристика домену	Приклади допустимих значень
Tel_No и Fax_No	Рядок змінної довжини, до 13 символів	0141-339-4439, 02124-67111
Property_No	Рядок змінної довжини, до 5 символів	PA14? PG4

Про виконану роботу студент має оформити. Робота повинна бути виконана на стандартних аркушах формату А4 у друкованому варіанті, обсягом 10-15 сторінок Набір здійснювати текстовим редактором Times New Roman, 14 кегль, міжрядковий інтервал 1,5, абзац 1,25 см. Рівняння тексту по ширині сторінки. Сторінки нумеруються в правому нижньому кутку арабськими цифрами без крапки наприкінці, додержуючись наскрізної нумерації. Першою сторінкою є титульна, на ній номер сторінки не проставляється. Практичні завдання передбачають створення бази даних засобами MS Access. Хід виконання завдань відображається у звіті. Результати виконання практичних завдань студент зобов'язаний подати в роздрукованому та електронному вигляді. Електроний варіант контрольної роботи повинен містити наступні файли: база даних MS Access, в якій зберігається результат виконання завдань; документ MS Word, який містить звіт про виконання контрольної роботи.

Перелік питань до заліку:

1. Що таке реляційна база даних?
2. Що таке відношення? Що таке ключ?
3. Для чого використовуються зв'язки між таблицями? Як здійснюється зв'язування таблиць?
4. Для чого використовується майстер підстановки?
5. Що визначає властивість поля "Присоединенный столбец"?
6. Що визначає властивість поля "Число столбцов"? Що визначає властивість поля "Ширина столбцов"?
7. Як визначити для підстановки поле зі списком з виведенням на екран двох стовбців?
8. Як приховати виведення стовпця ключа на екран при використанні підстановки?
9. Як визначити зв'язок між таблицями у MS Access?
10. Що таке каскадне відновлення у таблицях MS Access?

11. Що таке каскадне видалення даних у таблицях MS Access?
12. Дайте визначення першої, другої, третьої та четвертої нормальних форм.
13. Опишіть процедуру створення таблиці у MS Access.
14. Назвіть типи даних, які можна використовувати при визначенні таблиць у MS Access.
15. Як здійснити перевірку правильність значень, які користувач вводить у таблицю?
16. Як визначити повідомлення при введенні неправильного значення, яке користувач вводить у таблицю?
17. Що таке індекс? Як визначити індексоване поле?
18. Що таке запит? Які види запитів існують в MS Access?
19. Назвіть способи об'єднання таблиць у запитах.
20. Дайте визначення внутрішнього об'єднання таблиць.
21. Дайте визначення лівого, правого об'єднання таблиць
22. Наведіть приклади таблиць, для яких результат виконання лівого об'єднання не відповідатиме результату виконання внутрішнього об'єднання.
23. Наведіть приклади таблиць, для яких результат виконання правого об'єднання не співпадатиме результату виконання внутрішнього об'єднання.
24. Як створити запит на основі фільтра?
25. Що таке обчислюване поле? Правила побудови виразів у MS Access.
26. Побудова виразів засобами «Построителя выражений».
27. Використання вбудованих функцій у майстрі побудови виразів.
28. Як формуються прості умови відбору записів?
29. Як формуються складені умови відбору записів?
30. Як знайти унікальні значення деякого поля таблиці?
31. Які агрегатні функції можна використовувати в запитах?
32. Для чого використовується конструкція SELECT?
33. Для чого використовується службове слово AS?
34. Для чого використовується конструкція UNION?
35. Для чого використовується конструкція ORDER BY?
36. Для чого використовується конструкція GROUP BY?
37. Як мовою SQL визначається поле, за яким здійснюється об'єднання таблиць?
38. Як мовою SQL визначається умова відбору записів?
39. Як мовою SQL визначається групування записів?
40. Що таке запит з параметром?
41. Наведіть приклади запитів з використанням логічного оператора AND, OR.
42. Наведіть приклади запитів з використанням оператора BETWEEN.
43. Наведіть приклади запитів з використанням логічного оператора IN.
44. Наведіть приклади запитів з використанням логічного оператора LIKE.
45. Як в запитах здійснюється групування даних?

46. Опишіть створення перехресного запиту за допомогою майстра.
47. Як створити перехресний запит у режимі конструктора?
48. Опишіть створення запиту на пошук записів, що повторюються, за допомогою майстра.
49. Для чого використовуються запити-дії? Назвіть типи запитів дій.
51. Наведіть приклад запиту на створення таблиці.
52. Наведіть приклад запиту на додавання записів у таблицю.
53. Наведіть приклад запиту на видалення записів з таблиці.
54. Наведіть приклад запиту на оновлення записів у таблиці.
55. Для чого призначені форми? Способи створення форм у MS Access.
56. Для чого використовується заголовок форми? Для чого використовується примітка форми?
57. Для чого використовується сфера даних у формах?
58. Як створити форму з використанням полів кількох таблиць?
59. Як зв'язуються головна та підпорядкована форми?
60. Що таке підпорядкована форма? Опишіть способи створення підпорядкованих форм.
61. Для чого призначені елементи управління? Які елементи управління використовуються при створенні форм у MS Access?
62. Що таке кнопочні форми? Як створити кнопочну форму в MS Access?
63. Що таке діалогові форми? Як створити діалогову форму в MS Access?
64. Назвіть розділи звіту та опишіть їх призначення.

4.Завдання семестрових екзаменів:

Харківський національний університет імені В.Н. Каразіна

Факультет Міжнародних економічних відносин та туристичного бізнесу
Спеціальності:Міжнародна інформація
Спеціалізація _____
Семестр ___IV_____
Форма навчання _денна_____
Рівень вищої освіти (освітньо-кваліфікаційний рівень):__бакалавр_____
Навчальна дисципліна: __Основи організації бази даних_____

ЗАЛІКОВИЙ БІЛЕТ № 1

1 Що таке реляційна база даних? (15 балів).
2. Для чого використовується майстер підстановки? (15 балів)
3. Типи зв'язку між таблицями в реляційній моделі даних. (10 балів)

А) один до одного, один до багатьох, багато до одного
Б) один до одного
В) один до багатьох
Г) багато до одного
Д) немає вірної відповіді

Затверджено на засіданні кафедри міжнародних відносин, міжнародної інформації та безпеки.

Протокол № 1 від «28»серпня 2018 р.

Завідувач кафедри _____ (Л.В. Новікова)
підпис
Екзаменатор _____ (О.В. Чала)
підпис