Лекція

[**1.1. Інформаційна система**](https://rdb.dp.ua/uk/chapter_01#1.1)

[**1.2. Ручні картотеки – файлові системи – сучасні СКБД**](https://rdb.dp.ua/uk/chapter_01#1.2)

[**1.3. Порівняння СКБД файл-серверної архітектури з СКБД архітектури клієнт-сервер**](https://rdb.dp.ua/uk/chapter_01#1.3)

[**1.4. Життєвий цикл бази даних**](https://rdb.dp.ua/uk/chapter_01#1.4)

[**1.5. КОНТРОЛЬНІ ПИТАННЯ**](https://rdb.dp.ua/uk/chapter_01#1.5)

**Мета**

*Мета розділу – визначення ролі СКБД в інформаційній системі організації як її невід’ємного компонента, стислий розгляд різних технологій побудови СКБД, порівняння їх архітектур та ознайомлення із життєвим циклом баз даних.*

**1.1. Інформаційна система**

***База даних (БД)****– це сумісно використовуваний набір логічно зв'язаних даних та їх описів, призначений для задоволення інформаційних потреб організації* [4].

***Система керування базами даних (СКБД)****– це сукупність мовних і програмних засобів, які призначені для створення, ведення і сумісного використання БД багатьма користувачами.*

СКБД є невід'ємною частиною інформаційної системи організації [1-3].

***Інформаційна система****– ресурси, які дозволяють збирати, корегувати та поширювати інформацію для задоволення інформаційних потреб організації* [4].

До складу типової комп'ютеризованої інформаційної системи входять:

1. база даних;
2. програмне забезпечення баз даних;
3. прикладне програмне забезпечення;
4. апаратне забезпечення, у тому числі обладнання для зберігання даних;
5. персонал, що експлуатує й розробляє цю систему.

**1.2. Ручні картотеки – файлові системи – сучасні СКБД**

Надамо характеристику методам зберігання та керування даними, які використовуються інформаційними системами організацій [4].

***Ручні картотеки*** дозволяють успішно виконувати поставлені завдання обробки інформації, якщо кількість об'єктів, що зберігаються, невелика. Вони цілком придатні для одержання та зберігання великої кількості даних, але зовсім не підходять для тих випадків, коли необхідно встановити перехресні зв'язки або виконати обробку інформації. Якщо нам знадобиться якась інформація, то треба переглянути картотеку від початку й до кінця, щоб знайти потрібні дані. Для пошуку потрібної інформації передбачається використовувати в такій системі деякий алгоритм індексування, що дозволить прискорити пошук належних даних. Наприклад, можна використовувати спеціальні роздільники або окремі папки для різних типів об'єктів, що логічно пов’язані.

***Файлові системи*** – це набір програм, за допомогою яких користувач виконує деякі операції, наприклад, складає звіти. Кожна програма має свої власні дані і керує ними [2, 4, 5].

Файлові системи стали першою спробою комп'ютеризувати відомі всім ручні картотеки. Вони розглядалися згідно з потребами одержати більш ефективні засобів доступу до даних. Однак, замість створення централізованого сховища для всіх даних організації, був використаний децентралізований підхід, коли співробітники кожного відділу за допомогою фахівців з обробки даних працювали зі своїми власними даними й зберігали їх у своєму відділі.

Цього цілком достатньо, щоб зрозуміти існуючі недоліки:

1. розподіл і ізоляція даних;
2. дублювання даних;
3. залежність від даних;
4. несумісність форматів файлів;
5. швидке збільшення кількості прикладних програм для отримання інформації, яка необхідна користувачам.

***Розподіл та ізоляція даних*** призводить до істотних ускладнень, коли необхідно обробити інформацію, яка знаходиться у двох або більше файлах.

***Дублювання даних*** спричиняє неекономну витрату ресурсів, оскільки на уведення надлишкових даних потрібно витрачати додатковий час та кошти. Воно може призвести до порушення їх цілісності. Наприклад, у двох різних відділах організації можна одержати на одне й те саме питання суперечливі відповіді. У багатьох випадках дублювання даних можна уникнути сумісним використанням файлів.

***Залежність від даних*** стає причиною великих витрат при зміні фізичної структури файлів, яка відображена в програмному забезпеченні, що працює із ними. Наприклад, після деякого часу використання прикладної програми виявилося, що за зміненим законодавством номер банківського рахунку буде збільшений з 10 до 16 знаків. Щоб вийти із цієї ситуації, необхідно написати програму, яка відпрацює лише один раз. Вона повинна відкрити початковий файл, створити тимчасовий файл із новою структурою, зчитати інформацію з початкового файлу, перетворити дані в новий формат і записати їх у тимчасовий файл, вилучити початковий файл, присвоїти тимчасовому файлу його ім'я.

***Несумісність форматів файлів*** ускладнює обробку інформації ще в більшому обсязі, ніж розподіл та ізоляція даних. Наприклад, вона може бути наслідком необхідності спільного використання даних двома локальними задачами. Для цього знадобиться створювати програмне забезпечення, яке конвертує дані в один загальний формат. Після чого можлива їхня спільна обробка.

***Швидке збільшення кількості прикладних програм для отримання інформації***, яка необхідна користувачам файлових систем, багато в чому залежить від програмістів, оскільки усі потрібні запити й звіти повинні створювати саме вони. У результаті події розвивалися за одним із наступних сценаріїв. В організаціях, де програмісти були відсутні в штатному розкладі, типи створених запитів та звітів мали фіксовану форму, і не було ніяких інших інструментів для формування незапланованих або довільних запитів. В інших організаціях, де програмісти працювали, спостерігалося швидке збільшення кількості файлів та прикладних програм. Отже, настав час, коли вони перестали задовольняти потреби користувачів в одержанні необхідної інформації.

Зазначені обмеження файлових систем є наслідком двох факторів.

1. Визначення даних відбувається всередині прикладних програм, а не зберігається окремо й незалежно від них.
2. Крім прикладних програм не передбачено ніяких інших інструментів доступу до даних та їх обробки.

Для підвищення ефективності роботи стали використовувати новий підхід: БД і СКБД.

Щоб глибше уявити сутність цього поняття, розглянемо його більш детально. БД – це єдине, велике сховище даних, яке одноразово визначається, а потім використовується одночасно багатьма користувачами з різних підрозділів. Замість розрізнених файлів, тут усі дані зібрані разом з мінімальною часткою надмірності. БД уже не належить якомусь єдиному відділу, а є загальним корпоративним ресурсом. Причому в неї зберігаються не тільки робочі дані, але й їх опис. Тому її ще називають набором інтегрованих записів із самоописом. У сукупності опис даних називається системним каталогом (*system catalog*), або словником даних (*data-dictionary*), а самі елементи опису прийнято називати метаданими (*meta-data*), тобто "даними про дані". Саме наявність самоопису даних забезпечує незалежність прикладних програмам від даних (*program-data independents*).

На сьогодні використання СКБД має такі переваги:

1. контроль над надмірністю даних;
2. виключення суперечливості даних;
3. спільне використання даних;
4. підтримка цілісності даних;
5. підвищена безпека;
6. спрощення масштабування системи;
7. можливість знаходження компромісу при суперечливих вимогах;
8. підвищення доступності даних та їх готовності до роботи;
9. поліпшення показників продуктивності;
10. спрощення супроводу системи за рахунок незалежності від даних;
11. поліпшене керування паралельною обробкою даних;
12. розвинені служби резервного копіювання й оновлення,

та недоліки:

1. складність;
2. розмір;
3. вартість;
4. додаткові витрати на апаратне забезпечення;
5. витрати на конвертування даних;
6. більш серйозні наслідки при виході системи з ладу.

**1.3. Порівняння СКБД файл-серверної архітектури з СКБД архітектури клієнт-сервер**

**1.3.1. Основні складові програмного забезпечення СКБД**

***Сервер****це програмно-апаратний комплекс, який обробляє запит.*

Він, по суті, чекає, доки клієнт зробить запит, а потім обробляє його. Сервер має володіти здатністю обробляти одночасно кілька запитів від декількох клієнтів, а також уміти розподіляти ці запити за пріоритетами. Найчастіше серверна програма працює постійно для забезпечення безперервного доступу до її послуг.

***Клієнти****являють собою прикладні програми, що забезпечують графічний або інший інтерфейс із користувачем.*

Прикладні програми клієнтів надають користувачу інтерфейс для керування даними на сервері. Саме за допомогою прикладної програми користувач має доступ до функціональних можливостей сервера. Прикладом замовлених дій може бути одержання, додавання, корегування та вилучення інформації або друкованого звіту. У цьому випадку клієнт надсилає запит і надає необхідні для його виконання дані. Сервер відповідає за обробку запиту. Це означає, що клієнт може й самостійно виконувати будь-які логічні дії. Цілком можливо, що клієнт реалізує більшу частину (якщо не всі) бізнес-правила.

***Бізнес-правила****- це процедури керування, які визначають, як клієнт повинен одержувати доступ і маніпулювати даними на сервері.*

Ці правила реалізуються програмним текстом клієнта, сервера або ними обома. На боці сервера бізнес-правила реалізуються у вигляді збережуваних процедур, тригерів та інших об'єктів, які властиві серверній БД. Важливо розуміти, що бізнес-правила визначають поведінку всієї системи. За їх відсутності у нас є дані на одному комп'ютері й прикладна програма з інтерфейсом на іншому, однак немає методу їх з'єднання.

Якщо більша частина правил реалізована на сервері, то його називають "товстим сервером". Якщо правила реалізуються в основному на стороні клієнта, він називається "товстим клієнтом". Якщо правила реалізовані на середньому рівні, то сервер також називають "товстим". Отже, обсяг і тип операцій, які необхідні для керування даними, визначають місто реалізації бізнес-правил.

**1.3.2. Файл-серверна архітектура СКБД**

З часом БД на персональних комп'ютерах удосконалювалась за напрямом від настільних або локальних прикладних програм, коли реально із БД працювала тільки одна прикладна програма, до систем колективного доступу до даних [2 – 7].

Необхідність колективної роботи з тими самими даними сприяла перенесенню БД на файловий сервер. Прикладна програма, що працює з БД, розташовувалася також на сервері. Спосіб, що полягав у зберіганні прикладної програми, яка зверталася до БД, на комп'ютері користувача (клієнта) не набув широкого розповсюдження. Були розроблені нові версії локальних СКБД, які дозволяли створювати прикладні програми, що одночасно працюють з однією БД на файловому сервері. Основною сталася проблема, яка пов’язана з явною або неявною обробкою трансакцій, та забезпеченням змістовної та посилальної цілісності БД при одночасному корегуванні даних декількома клієнтами, що неминуче виникає при колективному доступі.

Під час експлуатації локальних систем були виявлені такі недоліки файл-серверного підходу:

1. Усі труднощі обчислювального навантаження при доступі до БД лягли на прикладну програму клієнта, що є наслідком принципу обробки інформації в системах "файл-сервер": при видачі запиту на вибірку інформації з таблиці вся таблиця БД копіюється на комп’ютер клієнта, де здійснюється вибірка.
2. Не оптимально витрачаються ресурси комп'ютера клієнта й мережі. Наприклад, якщо в результаті запиту ми маємо одержати 2 записи з файлу обсягом 10 000 записів, то усі вони будуть скопійовані з файл-сервера на комп'ютер клієнта. У результаті збільшується мережний трафік і зростають вимоги до апаратних потужностей системи. Помітимо, що потреби в постійному збільшенні обчислювальних потужностей комп'ютера клієнта обумовлюються не тільки удосконаленням програмного забезпечення, але й зростанням обсягів інформації, яка обробляється.
3. У БД на файл-сервері набагато простіше вносити зміни в окремі таблиці безпосередньо з інструментальних засобів, минаючи прикладні програми клієнтів. Подібна можливість виникає у зв'язку з тим, що у локальних системах база даних – поняття більше логічне, ніж фізичне, оскільки під БД розуміють набір окремих файлів, що знаходяться в єдиному каталозі на диску. Усе це дозволяє говорити про низький рівень безпеки як з погляду несанкціонованого доступу й нанесення шкоди, так і з погляду внесення помилкових змін.
4. Бізнес-правила в системах файл-сервера реалізуються в прикладних програмах. Це дозволяє їм застосовувати взаємовиключні бізнес-правила. Змістовна цілісність інформації при цьому може порушитися.

**1.3.3. Архітектура клієнт-сервер**

Наведені недоліки вирішуються при переході з архітектури "файл-сервер" [2, 4, 5] на архітектуру "клієнт-сервер", яка є наступним етапом у розвитку СКБД [2 – 7]. Характерна риса архітектури "клієнт-сервер" – перенесення обчислювального навантаження на сервер БД (*SQL*-сервер) і максимальне розвантаження прикладної програми клієнта від обчислювальних операцій, а також істотне підвищення рівня безпеки даних – як від зловмисних, так і просто помилкових змін. БД у цьому випадку міститься на сервері, як і в архітектурі "файл-сервep", однак прямого доступу до неї із прикладних програм не здійснює. Функції прямого звертання до БД виконує спеціальна керуюча програма – *SQL*-сервер, що є невід’ємною частиною СКБД.

Взаємодія сервера БД і прикладної програми клієнта здійснюється так: клієнт формує *SQL*-запит і надсилає його серверу. Сервер, який прийняв запит, виконує його і отриманий результат повертає клієнту. У прикладній програмі клієнта в основному відбувається інтерпретація отриманих від сервера даних, реалізується частина бізнес-правил та інтерфейс користувача для маніпулювання даними.

Багато із існуючих систем "клієнт-сервер" створювалися на базі прикладних програм, що працювали з локальними БД, які використовувалися спільно та були розміщені у мережах персональних комп'ютерів на файлових серверах. Перенесення на *SQL*-сервер СКБД файл-серверної архітектури здійснювалося з метою підвищення ефективності їх роботи, захищеності й надійності БД.

Модель архітектури "клієнт-сервер" передбачає наявність декількох рівнів. Дворівнева модель, можливо, найбільш загальна, оскільки вона додержується схеми побудови локальних БД (рис. 1.1). У цій моделі дані постійно знаходяться на сервері, а програмне забезпечення доступу до даних – на комп'ютері користувача. Бізнес-правила при цьому можуть розташовуватися на кожному з комп'ютерів (або навіть на обох одночасно).

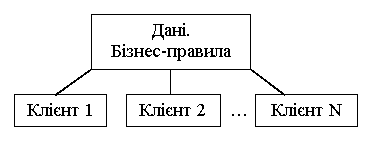


Рис. 1.1. Дворівнева модель "клієнт-сервер"

У трирівневій моделі архітектури "клієнт-сервер" (рис. 1.2) у програмному забезпеченні клієнта реалізований лише інтерфейс користувача, що дозволяє мати доступ до даних, які містяться на сервері. Прикладна програма клієнта здійснює запити для одержання доступу або зміни даних через сервер прикладних програм або сервер *Remote Data Broker* (Віддалений брокер-сервер даних) – *RDB*. Зазвичай бізнес-правила розташовуються саме на сервері *RDB*. Якщо клієнт, сервер і бізнес-правила розподілені по окремих комп'ютерах, розроблювач може оптимізувати доступ до даних і підтримувати їхню цілісність при звертанні до них з будь-яких прикладних програм у всій системі.

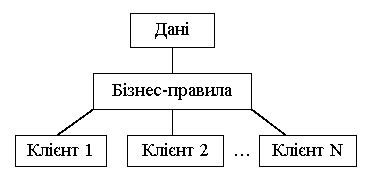


Рис. 1.2. Трирівнева модель 'клієнт-сервер'

Архітектура "клієнт-сервер" має ряд переваг:

1. Більшість обчислювальних процесів відбувається на сервері, що спрощує вимоги до обчислювальних потужностей комп'ютера клієнта.
2. Мінімізується мережний трафік за рахунок посилання сервером клієнту тільки тих даних, які він запитував. Наприклад, якщо необхідно зробити з файлу обсягом у 10 000 записів вибірку, результатом якої буде усього 2 записи, сервер виконає запит і перешле клієнту набір даних з 2 записів.
3. Спрощується процес зростання обчислювальних потужностей в умовах удосконалення програмного забезпечення і відповідно обсягів даних, що обробляються. Простіше й дешевше збільшити обчислювальну потужність сервера або замінити його на потужніший, ніж збільшувати потужності 100 – 500 комп'ютерів клієнтів.
4. БД краще захищена від несанкціонованого доступу, оскільки являє собою, як правило, єдиний файл, де знаходяться таблиці, обмеження цілісності та інші компоненти. Зламати таку БД, навіть за наявності наміру, досить важко. Значно збільшується її захищеність від уведення невірних значень, оскільки *SQL*-сервер автоматично перевіряє їх відповідність накладеним обмеженням та автоматично виконує бізнес-правила. Окрім того, сервер відслідковує рівні доступу для кожного користувача і блокує спроби виконати недозволені для нього дій. Усе це дозволяє говорити про високий рівень безпеки, що забезпечує посилальну й змістовну цілісність інформації.
5. Сервер керує трансакціями й запобігає спробам одночасної зміни одних й тих самих даних. Різні рівні ізоляції трансакцій дозволяють визначити поведінку сервера при виникненні ситуацій одночасної зміни даних.
6. Безпека системи зростає за рахунок перенесення більшої частини бізнес-правил на сервер. Падає питома вага взаємовиключних бізнес-правил у ПЗ клієнта, яке виконує різні операції над БД. У більшості випадків суперечливі дії будуть відхилені через автоматичне відстеження *SQL*-сервером посилальної цілісності даних.

**1.4. Життєвий цикл бази даних**

Життєвий цикл БД нерозривно пов'язаний із життєвим циклом інформаційної системи [1, 3, 4, 6, 8]. Він складається з таких етапів (рис. 1.3) [4]:

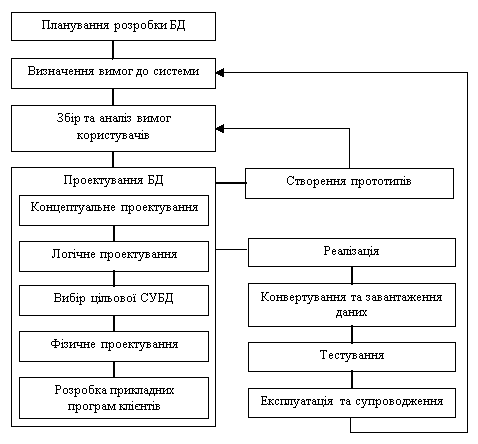


Рис. 1.3. Життєвий цикл БД

1. ***Планування розробки БД.*** Планування найефективнішого способу реалізації етапів життєвого циклу системи.
2. ***Формулювання вимог до системи.*** Визначення діапазону дій та обмежень БД, складу її користувачів та сфери застосування
3. ***Збір та аналіз вимог користувачів.*** На цьому етапі виконується збір і аналіз вимог користувачів з усіх можливих сфер застосування.
4. ***Проектування БД.*** Повний цикл розробки включає концептуальне, логічне й фізичне проектування БД, а також розробку прикладних програм клієнтів.
5. ***Розробка прикладних програм.*** Визначення інтерфейсу користувача, переліку й функцій прикладних програм, які працюють з БД.
6. ***Створення прототипів (необов'язково).*** Тут створюється робоча модель БД, яка дозволяє розробникам або користувачам уявити й оцінити остаточний вигляд і способи функціонування системи.
7. ***Реалізація.*** На цьому етапі в програмному коді реалізується концептуальна зовнішня й внутрішня функціональність БД і прикладного ПЗ, які визначені на попередніх етапах.
8. ***Конвертування й завантаження даних.*** Процес, який полягає в перетворенні та завантаженні даних (і прикладних програм) зі старої системи в нову.
9. ***Data converting and download.*** Data modification and download (also application programs) from an old system to a new one.
10. ***Тестування.*** БД тестується з метою виявлення помилок, а також її перевірки на відповідність усім вимогам, які висунуті користувачами.
11. ***Експлуатація та супровід.*** На цьому етапі БД вважається повністю розробленою і реалізованою. Надалі уся система буде перебувати під постійним спостереженням і відповідним чином підтримуватися. За потреби у прикладні програми, що функціонують, можуть вноситися зміни, які відповідають новим вимогам. Вони реалізуються за допомогою повторного виконання деяких етапів життєвого циклу, які наведені раніше.

Для малих БД з невеликою кількістю користувачів життєвий цикл може виявитися не дуже складним. При проектуванні великих БД, з десятками й навіть тисячами користувачів, сотнями запитів та прикладних програм, воно може стати занадто складним.

Слід відзначити, що розглянуті етапи не є надто послідовними, а містять деяку кількість повторів попередніх кроків у вигляді циклів зворотного зв'язку. Наприклад, при проектуванні БД можуть виникнути проблеми, для вирішення яких потрібно буде повернутися до етапу збору й аналізу вимог. Цикли зворотного зв'язку можуть існувати майже між усіма етапами. Ми розглянули найбільш явні з них.

**1.5. КОНТРОЛЬНІ ПИТАННЯ**

1. Що таке інформаційна система організації та з яких компонентів вона складається?
2. Як ручні картотеки пристосовані для зберігання та обробки даних?
3. Дайте визначення файловій системі з точки зору зберігання даних та керування ними.
4. До чого призводять відомі Вам обмеження файлових систем як сховищ даних?
5. Що таке база даних? Сформулюйте визначення.
6. Які переваги та недоліки має СКБД?
7. Порівняйте відомі Вам архітектури СКБД.
8. Які недоліки СКБД архітектури "файл-сервер" Вам відомі?
9. Які переваги СКБД архітектури "клієнт-сервер" Ви знаєте?
10. What advantages of “client-server” architecture DBMS you can list?
11. Дайте характеристику моделям архітектури "клієнт-сервер".
12. Яка існує функціональна різниця між серверною та клієнтською частинами СКБД архітектури "клієнт-сервер"?
13. Що таке бізнес-правила? Де вони реалізуються?
14. Що Ви знаєте про етапи життєвого циклу БД?

**Висновок**

*СКБД архітектури "  клієнт-сервер"   є невід'ємним компонентом сучасної інформаційної системи організації. Життєвий цикл БД не відокремлюється від життєвого циклу інформаційної системи. Його етапи та складність зворотних зв’язків залежать від напрямів та масштабу діяльності організації.*