# **Скорочення та умовні позначення**

**EOT** - end of transaction

**СУБД** – система управління базами даних

**ACID** - Atomicity, Consistency, Isolation, Durability

# **Розділ 1. Основні поняття**

**База даних** – структурована колекція даних, яка характеризує певну предметну область.

**Система управління базами даних (СУБД)** - програмне забезпечення, яке взаємодіє з користувачем, додатками та самою базою даних для збору та аналізу даних. СУБД дозволяє користувачеві взаємодіяти з базою даних. Дані, що зберігаються в базі даних, можуть бути змінені, вилучені та видалені. Вони можуть бути будь-яких типів, таких як рядки, числа, зображення, дата тощо.

**Предметна область** - це частина реального світу в межах певного контексту.

**Дані** – інформація (сукупність знань), придатна для обробки.

**Банк даних** – сховище даних, яке забезпечує незалежність даних від програми та зберігання, нагромадження, оновлення, пошук і видачу даних.

* До складу [**Банку даних**](#_Розділ_1._Основні) входять такі складові: сукупність технічного та програмного забезпечення, база даних, СУБД, словник даних, адміністратор БД.
* **Головне завдання БД** – гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином, БД складається з двох частин: збереженої інформації та системи керування нею.

На всіх етапах життєвого циклу БД складається з **двох** компонентів : **структури** та **даних**. На різних рівнях опису БД її структура зображується:

* на **інфологічному рівні** (сутність — зв’язок у ПО).
* на **датологічному рівні** (логічна модель - відображення логічних зв'язків між елементами даних безвідносно до їхнього змісту й середовищу зберігання).
* на **фізичному рівні** (записані дані; як саме вони зберігаються).

Використання принципів бази та банку даних передбачає організацію зберігання інформації у вигляді БД. Така організація даних усуває цілий ряд проблем:

1. відпадає потреба в кожній прикладній програмі детально вирішувати питання організації файлів;
2. усуває багаторазове введення й дублювання одних і тих самих даних;
3. не виникає проблеми зміни прикладних програм у зв’язку із заміною фізичних пристроїв або зміни структури даних;
4. підвищує рівень надійності та захищеності інформації;
5. зменшує надлишок даних.

**Адміністратор БД (АБД)** — особа або група осіб, які відповідають за загальне керування БД. Важливе завдання адміністратора БД — захист даних від злому, несанкціонованого та некомпетентного доступу.

**Таблиця** - організований набір даних у вигляді рядків (кортежів) та стовпців (атрибутів). **Кортеж** (поле) - це запис в таблиці. **Атрибут** (стовпець) - інформаційне відображення властивостей об'єкта.

**Існує два типи СУБД:**

* **Реляційна система** управління базами даних: дані зберігаються у відносинах (таблицях). Приклад - MySQL.
* **Нереляційна система** управління базами даних: немає поняття відносин, кортежів і атрибутів. Приклад - Mongo.
* **Підзапит** — це запит всередині іншого запиту, призначений для отримання даних або інформації з бази даних. У запиті зовнішній запит називається основним запитом, тоді як внутрішній запит називається підзапитом. Підзапити завжди виконуються першими, а результат підзапиту передається до основного запиту. Він може бути вкладений у SELECT, UPDATE або будь-який інший запит. Підзапит також може використовувати будь-які оператори порівняння, такі як >, < або =.
* **Сутності** : об'єкт у світі, дані про які можуть зберігатися у базі даних. У таблицях зберігаються дані, які є одним типом сутності. Наприклад, база даних банку має таблицю клієнтів для зберігання інформації про клієнтів. Таблиця клієнтів зберігає цю інформацію як набору атрибутів (стовпці в таблиці) кожного клієнта.
* **Відносини** : відносини чи зв'язки між сутностями, які мають якесь відношення одне до одного. Наприклад, ім'я клієнта пов'язане з номером облікового запису клієнта та контактною інформацією, яка може бути в тій же таблиці. Також можуть бути відносини між окремими таблицями (наприклад, клієнт до рахунків).

**Цілісність** даних визначає точність, а також узгодженість даних, що зберігаються у базі даних. Вона також визначає обмеження цілісності для забезпечення дотримання бізнес-правил для даних, коли вони вводяться в програму або базу даних.

Вимоги до реляційної СУБД :

1. Цілісність на рівні сутностей
2. Цілісність на рівні посилань

# **Розділ 2. Типи БД**

1. **Реляційні бази даних** (MySQL, Oracle DB, PostgreSQL) – найпопулярніший тип БД, у яких інформація зберігається як таблиць. У рядках знаходиться опис кожної окремої властивості об'єкта, а стовпці потрібні для отримання певних властивостей з рядка. Таблиці можуть взаємопов'язані. У реляційних БД є підтримка SQL, і навіть індексація, що дозволяє швидше знаходити необхідні дані. Особливий плюс таких баз - нормалізація даних: вони поділяються на різні таблиці, тому виключені повторювані або порожні комірки. Транзакції реляційних БД відповідають ACID - набору властивостей, що гарантує їхню надійну обробку. З мінусів баз можна відзначити відносно низьку швидкість доступу до даних, погану підтримку неструктурованих даних, складність масштабування та утворення великої кількості таблиць, через що важко зрозуміти структуру даних.
2. **Резидентні бази даних** (Redis, Apache Ignite, Tarantool) – Дані зберігаються в оперативній пам'яті. Дані обробляються швидко, тому резидентні БД є популярними там, де потрібно забезпечити максимально короткий час відгуку. Вони допомагають керувати телекомунікаційним обладнанням, проводити торги в онлайн-режимі або обслуговування Real-Time. Бази in-memory підтримують і швидке написання, і читання. В основному вони працюють із записами «ключ-значення», але також можуть працювати зі стовпцями. Щоб при несподіваному перезавантаженні не втратити дані, потрібно зробити запис із попереднім журналом на енергонезалежному пристрої. Це можна віднести до мінусів бази in-memory — доводиться вкладатися у дорогі інфраструктурні рішення, щоби забезпечити безперебійне харчування.
3. **Документоорієнтовані бази даних** (NoSQL : CouchDB, Couchbase, MongoDB) - Якщо реляційних БД для вилучення даних необхідно об'єднувати таблиці, то цих базах добре зберігається незв'язана інформація у великих обсягах. Вони підтримують JSON. Для будь-якого ключа можна створити складне значення і одразу включити всю структуру даних до одного запису. Вибірка на запит може містити частини безлічі документів без їх повного завантаження в оперативну пам'ять. У документоорієнтованих базах немає прив'язки до схеми. Вони підходять для OLTP та підтримують складні типи. Такі БД вважають за краще використовувати в системах управління вмістом, для пошуку документів, у видавничій справі. Три недоліки бази даних — відсутність хорошої аналітичної підтримки та підтримки транзакцій, а також складнощі з масштабуванням.
4. **Пошукові бази даних** (Elastic) - Цей тип БД необхідний отримання відомостей через фільтр. Шукати можна за будь-яким введеним значенням, у тому числі за окремими словами. Можна скористатися повнотекстовим пошуком. Пошукові бази даних добре масштабуються та зручні для зберігання журналів, об'ємних текстових значень. Можна використовувати пошукові БД для моніторингу оптимізації цін, виявлення помилок у додатку з бронювання квитків та вирішення багатьох інших завдань. У основі можуть зберігатися мільярди документів. Пошук здійснюється швидко. Мінуси системи - погана аналітична підтримка та обмежена можливість застосування БД (можна використовувати лише для пакетних вставок).
5. **Бази даних із широкими стовпцями** (Cassandra, Google BigTable, HBase) - БД з широкими стовпцями можуть вимагати більші обсяги даних швидше, ніж звичайні реляційні. Відомості зберігаються у вигляді записів «ключ-значення» на жорсткому диску або твердому накопичувачі. Бази даних з широкими стовпцями дозволяють виконувати швидкий запис рядковим і швидким читанням по ключу. БД добре масштабуються та підходять для організації магазинних каталогів, механізмів виявлення шахрайства. Їх зручно використовуватиме управління величезними обсягами інформації на безлічі загальних серверів у розподіленій системі. Недоліками бази даних є те, що вона працює у форматі «ключ-значення» і не має підтримки аналітики.
6. **Стовпчасті бази даних** (Clickhouse, Vertica) - У БД такого типу дані зберігаються у стовпцях, а чи не в рядках. Доступ до вмісту здійснюється без допомоги ключів. При використанні стовпчастих баз даних використовують пакетну вставку, щоб можна було готувати інформацію для швидкого читання стовпчиками. У стовпчастих БД є підтримка аналітики та можливість зручного масштабування. Такі бази даних використовують там, де потрібно запитувати інформацію щодо певних стовпців, — у системах роздрібного продажу та фінансових транзакцій. Основний мінус у БД лише один: вона підходить лише для пакетних вставок.
7. **Графові бази даних** (OrientDB, Neo4j) - Дані зберігаються у вигляді графів, тобто моделей із вузлами та зв'язками. Вони досить гнучкі, з логічною структурою. Вузли служать для зберігання сутностей даних, а ребра - для збереження взаємозв'язків між сутностями, якими можна керувати. Графові БД застосовують вирішення завдань у біоінформатиці, і навіть для моделювання соціальних мереж, щоб зберігати взаємозалежну інформацію людей. Бази даних такого типу погано піддаються масштабуванню, а другий їх недолік - необхідність використовувати особливу мову запитів SPARQL, яка відрізняється від SQL.

* Вимоги **ACID** - набір вимог, які забезпечують збереження даних.
  + **A**[**tomicity** - Атомарність](https://habr.com/ru/post/555920/#atomicity)

Атомарність гарантує, що кожну транзакцію буде виконано повністю або не буде виконано зовсім. Не допускаються проміжні стани.

* + **Consistency** – Узгодженість

Ця властивість випливає із попереднього. Завдяки тому, що транзакція не допускає проміжних результатів, база залишається консистентною. У класичному сенсі ця властивість означає, що транзакція може бути успішно завершена з фіксацією результатів своїх операцій тільки в тому випадку, коли дії операцій не порушують цілісність бази даних, тобто задовольняють набору обмежень цілісності, визначених для цієї бази даних.

* + **Isolation** – Ізольованість

Потрібно, щоб дві одночасно виконувані транзакції жодним чином не діяли одна на іншу. Іншими словами, результати виконання операцій транзакції **T1** не повинні бути видні ніякий іншій транзакції **T2** до тих пір, поки транзакція **T1** не завершується успішно. Спосіб вирішення : **Блокування в СУБД** - відмітка про захоплення об'єкта транзакцією в обмежений або виключний доступ з метою запобігання колізії та підтримки цілісності даних .

* + **Durability** – Надійність.

Після успішного завершення транзакції всі зміни, які були внесені в стан бази даних операціями цієї транзакції, повинні гарантовано зберігатися, навіть у разі збоїв апаратури або програмного забезпечення.

**12 правил Кодда** - 12 правил, насправді їх 13, яким має задовольняти кожна система управління реляційними базами даних.

# **Розділ 3. Нормалізація, декомпозиція**

**Нормалізація** – організація даних шляхом розбиття таблиць відповідно до алгоритму нормалізації на відношення на базі функціональних залежностей. Процес – видалення надлишкових даних

**Денормалізація** — техніка, яка використовується для перетворення з найвищих до нижчих нормальних форм. Вона допомагає розробникам баз даних підвищити продуктивність усієї інфраструктури, оскільки вносить надмірність до таблиці. Вона додає надлишкові дані до таблиці, враховуючи часті запити до бази даних, які поєднують дані з різних таблиць в одну таблицю.

**Нормальна форма** – сукупність вимог, яким має відповідати БД.

Переваги нормалізації :

1. Краща організація даних
2. Більш гнучкі запити
3. Проста модифікація
4. Швидкий пошук
5. Скорочення надлишковості та дублювання даних

**Надлишковість** – зберігання більшого обсягу даних, аніж цього потрібно. Призводить до аномалій.

**Аномалія в БД** – проблеми, що виникають з даними через дефекти у проектуванні БД.

Типи аномалій :

1. **Оновлення** – зміна даних в 1-ій комірці змушує до зміни даних у всій таблиці.
2. **Додавання** – додавання небажаної чи неіснуючої інформації про певну сутність.
3. **Видалення** – втрата даних при видаленні інших даних, які можуть бути не пов’язані.

**Декомпозиція** – розділення об’ємного відношення на менші самодостатні та взаємопов’язані відношення за правилами, що задовольняють НФ.

Нормальні форми :

1. **1 НФ** – немає повторюваних чи більше 1 значень у стрічці.
2. **2 НФ** – відношення у **1 НФ** і кожен не ключовий атрибут залежить від PK.
3. **3 НФ** – відношення у **2 НФ** і кожне не ключ. Значення залежить від PK і не має залежності від іншого не ключового значення.

# **Розділ 4. Ключі, обмеження, індекси, представлення, тригери**

Ключі :

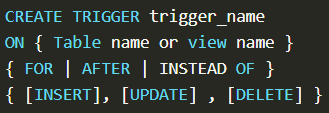
1. **Первинний** (PK) – стовпець, що має значення запису та унікального визначає його. Не може бути NULL та не може повторюватись. Найшвидший індекс. Може бути простий та складений. Autoincrement = IDENTITY(1,1) – MS SQL; AUTO\_INCREMENT - MySQL.
2. **Зовнішній** (FK) – значення посилається на PK іншої таблиці.

**Індекс** – структура даних, що швидко визначає розташування певних даних у базі. Створюється для стовпця з метою підвищення продуктивності запитів. Реалізується автоматично СУБД. Унікальний індекс – не дозволяється дублювання.

* + При змінах в таблиці відповідні зміни мають бути проведені з індексами.
* **Кластерний індекс** – це тип індексу, який сортує рядки даних у таблиці за їх ключовими значеннями, тоді як **некластеризований** індекс зберігає дані в одному місці та індекси в іншому місці.

**Представлення** –віртуальна таблиця, заснована на результаті запиту з таблиці, з якої беруться посилання на дані.

**Тригери** – об’єкт, тип збережуваної процедури, який реагує на ті чи інші дії у БД.



**Транзакція** - логічна одиниця роботи, яка містить один або більше операторів SQL. Транзакція є атомарною одиницею. Наслідки всіх інструкцій SQL у транзакції можуть бути або 100% зафіксованими або скасованими з бази даних.

Обмеження :

1. **NOT NULL** – не пуста комірка
2. **UNIQUE** – унікальне значення в даній комірці (дозволяється NULL 1 раз)
3. **DEFAULT** – стандартне значення при відсутності даних
4. **PK** – primary key
5. **FK** – foreign key
6. **CHECK** – перевіряє, чи в даному стовпці задовольняється умова

# **Розділ 5. Підмножини, оператори SQL, зв’язки**

* **DDL (Data Definition Language, мова опису даних)** – дозволяє виконувати різні операції з базою даних, такі як CREATE (створення), ALTER (зміна) та DROP (видалення об'єктів).
* **DML (Data Manipulation Language, мова керування даними)** – дозволяє отримувати доступ до даних та маніпулювати ними, наприклад, вставляти, оновлювати, видаляти та вилучати дані з бази даних.
* **DCL (Data Control Language, мова контролю даних)** - дозволяє контролювати доступ до бази даних. Приклад - GRANT (надати права), REVOKE (відкликати права).

У SQL доступно три типи оператора, а саме:

1. Арифметичні оператори
2. Логічні Оператори
3. Оператори порівняння

Типи :

* **CHAR(size)** – фіксований рядок на певну кількість символів.
* **VARCHAR(size)** – рядок зі змінною кількістю символів.

Там, де n на початку – усі символи з Юнікоду.

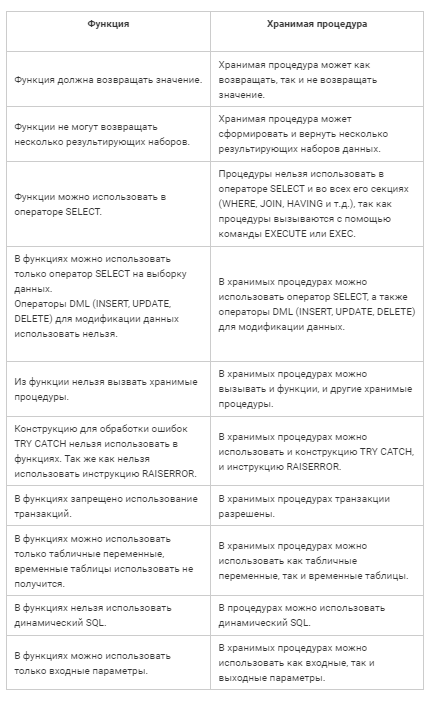
* **INT** – ціле число.
* **BIGINT** – ціле число (аналог long).
* **MONEY** – для грошей (8 байт).
* **FLOAT(N)** – при n=24 – 4 байти; при n=53 – 8 байт.
* **DATETIME, DATE, TIME** – взаємодія з часом (8, 3, 3-5 байт)

Зв’язки :

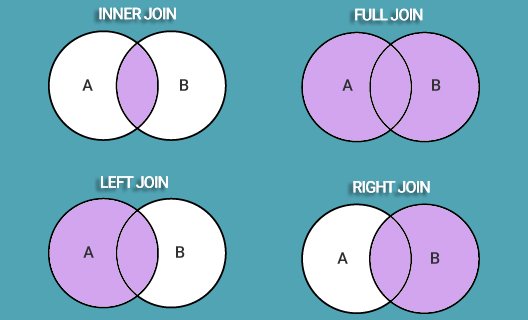
1. **Один до одного** – кожному запису в таблиці А буде відповідати лише один пов'язаний з ним запис в таблиці Б.
2. **Один до багатьох** – в одному відділі може працювати багато працівників, але жоден працівник не може числитись одразу в кількох відділах.
3. **Багато до багатьох** – певний відділ може займати приміщення на різних поверхах, а на одному поверсі можуть розміщуватися одразу декілька відділів.



# **Питання**



Joins :



**INNER JOIN —** отримання записів з одинаковими значеннями в об’єднаних таблицях, т.е. получение пересечения таблиці.

**FULL OUTER JOIN** — об’єднує записи з об’єднаних таблиць (якщо умова об’єднання рівна true) і доповнює їх усіма записами з об’єднаних таблиць, які не мають совпадінь. Для записів, які не мають зіставлення з іншої таблиці, недостаюче поле матиме значення NULL.

**LEFT JOIN** — повертає всі записи, що відповідають умовам об’єднання, плюс усі записи, що залишилися із зовнішньої (лівої) таблиці, які не відповідають умовам об’єднання.

**RIGHT JOIN** — працює точно так само, як і ліве об’єднання, тільки в якості зовнішньої таблиці буде використовуватися права.

**DELETE && TRUNCATE**

Об'єднайте різницю між командами DELETE і TRUNCATE

Команда DELETE — це DML-операція, яка видаляє записи з таблиці, відповідному заданому умові:

При цьому створюються логи видалення, то цю операцію можна видалити.

Ось команда TRUNCATE — це DDL-операція, яка повністю перетворює таблицю, і видалити таке видалення неможливо:

**Вопрос 12. Що розуміється під цілістю даних?**

Цілість даних визначає точність, а також узгодженість даних, що зберігаються в базі даних. Вона також визначає обмеження повноцінності для забезпечення збереження бізнес-правил для даних, коли вони вводяться в додаток або базу даних.

Під узгодженістю розуміється, що цілісність бази даних не порушиться, незважаючи на те, чи була підтверджена транзакція, чи відкинута.

**Питання 13. У чому різниця між кластеризованими та некластеризованими індексами в SQL?**

Різниці між кластеризованими та некластеризованими індексами в SQL:

Кластерний індекс використовується для простого та швидкого отримання даних із бази даних, тоді як чтення з некласстеризованого індексу відбувається відносно повільніше.

Кластеризований індекс змінює спосіб зберігання записів у базі даних — він сортує рядки за столбцем, який встановлено як кластеризований індекс, тоді як у некласстеризованому індексі він не змінює спосіб зберігання, але створює окремий об’єкт усередині таблиці, який вказує на вихідні рядки таблиці при пошуку.

Одна таблиця може мати тільки один кластеризований індекс, тоді як некластеризованих у нього може бути багато.

**Питання 21. У чому розниця між командами DROP і TRUNCATE?**

Команда DROP видаляє саму таблицю, і не можна виконувати команди Rollback, тоді як команда TRUNCATE видаляє всі рядки з таблиці (прим. переклад: в SQL Server Rollback нормально працює і відкатує DROP).

**Вопрос 27. У чому розниця між перехресним (перехресним) і природним (природним з'єднанням) з'єднанням?**

Перехрестне з'єднання створює перекрестне або декартове твори двох таблиць, тоді як природне з'єднання засновано на всіх столбцях, які мають одне ім'я та типи даних в інших таблицях.

**10. Питання: Ви можете пояснити типи запитів?**

Відповідь: підзапити бувают 3 типів:

**Однострічковий підзапит** - повертає тільки одну строку.

**Підзапит с декількома строками** - повертає кілька строк.

**Підзапит с декількома атрибутами** - возвращает несколько столбцов.

14. Питання: Які функції є у SQL Server?

**Табличні функції**

**Скалярні функції**

**Агрегатні функції**

**Системні функції**

Агрегатні функції

Функції конфігурації

Функції курсору

Функції даних та часу

Математичні функції

Функції метаданих

Інші функції

Функції ідентифікатора ієрархії

Функції набору рядків

Функції безпеки

Рядкові функції

Системні статистичні функції

Текстові та графічні функції

**17. Питання: Розрізняйте команди UNION та UNION ALL.**

Відповідь: Команда UNION використовується для вибору аналогічної інформації із двох таблиць. Команда UNION ALL робить те саме, але вибирає всі доступні значення і не видаляє рядки, що повторюються.

**24. Питання: Що ви маєте на увазі під тригерами в SQL Server?**

Відповідь: У SQL Server тригери - це об'єкти бази даних, схожі на процедури, що зберігаються. Ці спеціальні типи процедур, що зберігаються, запускаються, коли у відповідних базах даних відбувається певна подія. Тригери бувають трьох типів:

**Тригер DDL** – спрацьовує у відповідь на події команди DDL, починаючи з Alter, Create та Drop.

**Тригер DML** – спрацьовує у відповідь на події команди DML, що починаються з команд Delete, Insert, Update.

**ВИДИ ІНДЕКСІВ**

**Кластеризований** індекс зберігає реальні дані, не можна створити більше одного **кластеризованого** індексу у таблиці.

**Некластеризований** індекс не містить реальних даних таблиці у вузлах-листках. Вузли-листя містять один із двох типів інформації про місцезнаходження рядків даних.

**32. Питання: Що таке денормалізація у базах даних?**

Відповідь: Денормалізація - це процес навмисного запровадження надлишкових даних у таблиці бази даних для оптимізації продуктивності. У цьому процесі покращення продуктивності читання замінюється деяким погіршенням продуктивності запису.

PS – Слід зазначити, що ненормалізовані та денормалізовані бази даних – це зовсім різні поняття. Ненормалізована база даних - це база даних, яка була нормалізована. З іншого боку, денормалізована база даних - база даних, яка спочатку нормалізується, та був денормалізується.