**Лекція №7. Мова SQL. Загальний опис.**

**Стандарт мови запитів SQL**

У реалізаціях конкретних реляційних БД зараз не використовується в чистому вигляді ні реляційна алгебра, ні реляційне числення. Фактичним стандартом доступу до реляційних даних стала мова SQL (Structured Query Language). Мова SQL являє собою суміш операторів реляційної алгебри і виразів реляційного числення, що використовує синтаксис, близький до фраз англійської мови і розширений додатковими можливостями, відсутніми в реляційної алгебри та реляційному численні. Взагалі, мова доступу до даних називається реляційно повним, якщо він по виразною силою не поступається реляційної алгебри (або, що-те ж саме, реляційному обчисленню), тобто будь-який оператор реляційної алгебри може бути виражений засобами цієї мови. Саме таким і є мова SQL.

*SQL ( Structured Query Language )* – Структурована Мова Запитів - стандартна мова запитів по роботі з реляційними БД. Робота була розпочата відразу після появи статті Е.Кодда в 1970р. в лабораторіях компанії IBM для перевірки можливостей реляційної моделі. Основні віхи становлення і розвитку мови SQL:

СУБД System R - експериментальна дослідницька система з мовою SEQUEL (пізніше SQL), створена IBM:

SQL в комерційних реалізаціях:

– 1979 - Oracle (Relation Software Inc.- Oracle corp.;

– 1981-1982 - DB2 (IBM), Ingres - CA-OpenIngres (Relation Technology Inc. - Computer Associates);

– 1984 - Informix (Informix Sofrware);

– 1986 - Sybase (Sybase Corp.).

Стандартизація SQL:

– міжнародний стандарт 1989 р (SQL 1);

– міжнародний стандарт 1992 р (SQL 2) ;

– стандарт 1998 р (SQL 3).

Повністю історія версій стандартів-ревізій SQL:

| **Рік** | **Назва** | **Інша назва** | **Коментар** |  |
| --- | --- | --- | --- | --- |
| [1986](https://uk.wikipedia.org/wiki/1986) | [SQL-86](https://uk.wikipedia.org/w/index.php?title=SQL-87&action=edit&redlink=1) | SQL-87 | Вперше оприлюднено [ANSI](https://uk.wikipedia.org/wiki/ANSI). Ратифіковано [ISO](https://uk.wikipedia.org/wiki/ISO) в [1987](https://uk.wikipedia.org/wiki/1987). |  |
| [1989](https://uk.wikipedia.org/wiki/1989) | [SQL-89](https://uk.wikipedia.org/w/index.php?title=SQL-89&action=edit&redlink=1) | FIPS 127-1 | Незначні зміни. |  |
| [1992](https://uk.wikipedia.org/wiki/1992) | [SQL-92](https://uk.wikipedia.org/wiki/SQL-92) | SQL2 | Вагомі зміни. |  |
| [1999](https://uk.wikipedia.org/wiki/1999) | [SQL:1999](https://uk.wikipedia.org/wiki/SQL1999) | SQL3 | Додано [регулярні вирази](https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%BD%D1%96_%D0%B2%D0%B8%D1%80%D0%B0%D0%B7%D0%B8), рекурсивні запити, тригери та деякі [об'єктно-орієнтовані](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) нововведення. |  |
| [2003](https://uk.wikipedia.org/wiki/2003) | [SQL:2003](https://uk.wikipedia.org/wiki/SQL2003) | SQL 2003 | Впроваджені розширення для роботи з [XML](https://uk.wikipedia.org/wiki/XML)-даними. |  |
| [2006](https://uk.wikipedia.org/wiki/2006) | [SQL:2006](https://uk.wikipedia.org/wiki/SQL:2006) | SQL 2006 | ISO/IEC 9075-14:2006. Функціональність роботи з XML-даними значно розширено. З'явилась можливість сумісного використання в SQL та [XQuery](https://uk.wikipedia.org/wiki/XQuery). |  |
| [2008](https://uk.wikipedia.org/wiki/2008) | [SQL:2008](https://uk.wikipedia.org/wiki/SQL:2008) | SQL 2008 | Вдосконалені можливості віконних функцій, усунуто деякі неоднозначності стандарту SQL:2003. Легалізовано ORDER BY поза визначенням курсору. Додано тригери INSTEAD OF. Додано заяви TRUNCATE. |  |
| [2011](https://uk.wikipedia.org/wiki/2011) | [SQL:2011](https://uk.wikipedia.org/wiki/SQL:2011) | SQL 2011 | Додає часові дані (PERIOD FOR) (додаткова інформація у: [Тимчасова база даних#Історія](https://uk.wikipedia.org/w/index.php?title=%D0%A2%D0%B8%D0%BC%D1%87%D0%B0%D1%81%D0%BE%D0%B2%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85&action=edit&redlink=1)). Покращення функцій вікон та пропозиції FETCH. |  |
| [2016](https://uk.wikipedia.org/wiki/2016) | [SQL:2016](https://uk.wikipedia.org/wiki/SQL:2016) | SQL 2016 | Додає рядки підрівнювання посилань, поліморфні функції таблиці, JSON. |  |

Текст стандарту SQL 2 займає 600 станиць сухого формального тексту, це дуже багато, і здається, що це просто підступи розробників стандартів, а не те, що необхідно, рядовим розробникам. Однак жоден серйозний розробник, який працює з базами даних, не повинен ігнорувати стандарт, оскільки стандарти - це вірний орієнтир для розробників, так як всі постачальники СУБД в своїх перспективних розробках обов'язково слідують стандарту, і можна бути впевненим, що стандарт буде реалізований практично у всіх перспективних СУБД. Так сталося з кожним стандартом SQL 1, SQL 2, SQL 3.

З використанням будь-яких стандартів зв'язані не лише багаточисельні і очевидні переваги, але і певні недоліки. Перш за все, стандарти направляють в певне русло розвиток відповідної індустрії; зокрема мова наявність стандарту SQL приводить до сумісності його різних реалізацій і сприяє як підвищенню переносимості програмного забезпечення і баз даних в цілому, так і універсальності роботи адміністраторів баз даних. З іншого боку, стандарти обмежують гнучкість і функціональні можливості конкретної реалізації. Під реалізацією мови SQL розуміється програмний продукт SQL відповідного виробника. Для розширення функціональних можливостей багато розробників, що дотримуються прийнятих стандартів, додають до стандартної мови SQL різні розширення. Стандарти вимагають від будь-якої закінченої реалізації мови SQL наявності певних характеристик і у загальних рисах відображають основні тенденції, які не лише наводять до сумісності між всіма конкуруючими реалізаціями, але і сприяють підвищенню значущості програмістів SQL і користувачів реляційних баз даних на сучасному ринку програмного забезпечення. Всі конкретні реалізації мови декілька відрізняються один від одного.

На користь самих же виробників гарантувати, аби їх реалізація відповідала сучасним стандартам ANSI в частині переносимості і зручності роботи користувачів. Проте, кожна реалізація SQL містить удосконалення, що відповідають вимогам того або іншого сервера баз даних. Ці удосконалення або розширення мови SQL є додатковими командами і опціями, що є додаваннями до стандартного пакету і доступні в даній конкретній реалізації.

Сьогодні мова SQL підтримується багатьма десятками СУБД різних типів, розроблених для найрізноманітніших обчислювальних платформ, починаючи від персональних комп'ютерів і закінчуючи мейнфреймами.

Всі мови маніпулювання даними, створені для багатьох СУБД до появи реляційних баз даних, були орієнтовані на операції з даними, представленими у вигляді логічних записів файлів. Зрозуміло, це вимагало від користувача детального знання організації зберігання даних і серйозних зусиль для вказівки того, які дані необхідні, де вони розміщуються і як їх отримати.

Мова SQL орієнтована на операції з даними, представленими у вигляді логічно взаємозв'язаних сукупностей таблиць. Найважливіша особливість його структур – орієнтація на кінцевий результат обробки даних, а не на процедуру цієї обробки. Мова SQL сам визначає, де знаходяться дані, індекси і навіть які найбільш ефективні послідовності операцій слід використовувати для здобуття результату, а тому вказувати ці деталі в запиті до бази даних не потрібно

**Типи команд SQL**

Реалізація в SQL концепції операцій, орієнтованих на табличне представлення даних, дозволила створити компактну мову з невеликим набором пропозицій. Мова SQL може використовуватися як для виконання запитів до даних, так і для побудови прикладних програм. Основні категорії команд мови SQL призначені для виконання різних функцій, включаючи побудову об'єктів бази даних і маніпулювання ними, початкове завантаження даних в таблиці, оновлення і видалення існуючої інформації, виконання запитів до бази даних, управління доступом до неї та її загальне адміністрування.

Основні категорії команд мови SQL:

– DDL – мова визначення даних;

– DML – мова маніпулювання даними;

– DQL – мова запитів;

– DCL – мова управління даними;

– команди адміністрування даних;

– команди управління транзакціями.

Визначення структур бази даних (DDL)

На відміну від реляційної алгебри, де були представлені тільки операції запитів до БД, SQL є повною мовою, в ньому присутні не тільки операції запитів, але і оператори, відповідні DDL - Data Definition Language - мови опису даних. Крім того, мова містить оператори, призначених для управління (адміністрування) БД.

Мова визначення даних (Data Definition Language, DDL) дозволяє створювати і змінювати структуру об'єктів бази даних, наприклад, створювати і видаляти таблиці. Основними командами мови DDL є наступні: CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX, ALTER INDEX, DROP INDEX.

Маніпулювання даними (DML)

Мова маніпулювання даними (Data Manipulation Language DML) використовується для маніпулювання інформацією усередині об'єктів реляційної бази даних за допомогою трьох основних команд: INSERT, UPDATE, DELETE.

Вибірка даних (DQL)

Мова запитів DQL найбільш відома користувачам реляційної бази даних, не дивлячись на те, що він включає одну команду: SELECT. Ця команда разом зі своїми багаточисельними опціями і пропозиціями використовується для формування запитів до реляційної бази даних.

Мова управління даними (DCL)

Команди управління даними дозволяють управляти доступом до інформації, бази даних, що знаходиться усередині. Як правило, вони використовуються для створення об'єктів, пов'язаних з доступом до даних, а також служать для контролю над розподілом привілеїв між користувачами. Команди управління даними наступні: GRANT, REVOKE.

Команди адміністрування даних

За допомогою команд адміністрування даних користувач здійснює контроль за виконуваними діями і аналізує операції бази даних; вони також можуть виявитися корисними при аналізі продуктивності системи. Не слід плутати адміністрування даних з адмініструванням бази даних, яке є загальним управлінням базою даних і має на увазі використання команд всіх рівнів.

SQL містить розділи, представлені в таблиці 1.

Таблиця 1 – SQL розділи

| **Оператор** | **Суть** | | | | **Дія** | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Оператори визначення даних DDL* | | | | | | | | | | | |
| CREATE TABLE | створити таблицю | | | | Створює нову таблицю в БД | | | | | | |
| DROP TABLE | видалити таблицю | | | | Видаляє таблицю з БД | | | | | | |
| ALTER TABLE | змінити таблицю | | | | Змінює структуру існуючої таблиці або обмеження цілісності, що задаються для даної таблиці | | | | | | |
| CREATE VIEW | створити уявлення | | | | Створить віртуальну таблицю, відповідну деякого SQL-запит | | | | | | |
| ALTER VIEW | змінити уявлення | | | | Змінює раніше створене подання | | | | | | |
| DROP VIEW | видалити уявлення | | | | Видаляє раніше створене подання | | | | | | |
| CREATE INDEX | створити індекс | | | | Створює індекс для деякої таблиці для забезпечення швидкого доступу по атрибутам, що входять в індекс | | | | | | |
| DROP INDEX | видалити індекс | | | | Видаляє раніше створений індекс | | | | | | |
| *Оператори маніпулювання даними Data Manipulation Language* | | | | | | | | | | | |
| DELETE | Видалити рядки | | | | Видаляє одну або кілька рядків, відповідних умовам фільтрації, з базової таблиці. Застосування оператора узгоджується з принципами підтримки цілісності, тому цей оператор не завжди може бути виконаний коректно, навіть якщо синтаксично він записаний правильно | | | | | | |
| SELECT | Вибрати рядки | | | | Оператор, який замінює всі оператори реляційної алгебри і дозволяє сформувати результуюче відношення, відповідне запитом | | | | | | |
| *Засоби управління транзакціями* | | | | | | | | | |
| COMMIT | | | | Завершити транзакцію | | Завершити комплексну взаємозв'язану обробку інформації, об'єднану в транзакцію | | | |
| ROLLBACK | | | | Відкотити транзакцію | | Скасувати зміни, проведені в ході виконання транзакції | | | |
| SAVEPOINT | | | | Зберегти проміжну крапку виконання транзакції | | Зберегти проміжний стан БД, позначити його для того, щоб можна було в подальшому до нього повернутися | | | |
| *Засоби адміністрування даних* | | | | | | | | | |
| ALTER DATABASE | | | | змінити БД | | Змінити набір основних об'єктів в базі даних, обмежень, що стосуються бази даних | | | |
| ALTER DBAREA | | | | Змінити область зберігання БД | | Змінити раніше створену область зберігання | | | |
| ALTER PASSWORD | | | | змінити пароль | | Змінити пароль для всієї бази даних | | | |
| CREATE DATABASE | | | | створити БД | | Створити нову базу даних, визначивши основні параметри для неї | | | |
| CREATE DBAREA | | | | Створити область зберігання | | Створити нову область зберігання і зробити її доступною для розміщення даних | | | |
| DROP DATABASE | | | | видалити БД | | Видалити існуючу базу даних (тільки в тому випадку, коли ви маєте право виконати цю дію) | | | |
| DROP DBAREA | | | | Видалити область зберігання БД | | Видалити існуючу область зберігання (якщо в ній на зараз не розташовуються активні дані) | | | |
| GRANT | | | | надати права | | Надати права доступу на ряд дій над деякими об'єктом БД | | | |
| REVOKE | | | | позбавити прав | | Позбавити прав доступу до деякого об'єкту або деяких дій над об'єктом | | | |
| *Програмний SQL* | | | | | | | | |
| DECLARE | | | Визначає курсор для запиту | | | | | Задає деякий ім'я і визначає пов'язаний з ним запит до БД, який відповідає віртуального набору даних | | | | |
| OPEN | | відкрити курсор | | | | | Формує віртуальний набір даних, відповідає опису зазначеного курсору і поточному стану БД | | | |
| FETCH | | Вважати рядок з безлічі рядків, визначених курсором | | | | | Зчитує чергову рядок, задану параметром команди з віртуального набору даних, відповідає відкритому курсору | | | |
| CLOSE | | Закрити курсор | | | | | Припиняє доступ до віртуального набору даних, відповідному зазначеним курсору | | | |
| PREPARE | | Підготувати оператор SQL до динамічного виконання | | | | | Згенерувати план виконання запиту, відповідного заданому оператору SQL | | | |
| EXECUTE | | Виконати оператор SQL, раніше підготовлений до динамічного виконання | | | | | Виконує раніше підготовлений план запиту | | | |

**Структура мови SQL. Типи даних.**

**Типи 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 найзагальнішому вигляді домен визначається завданням деякого базового типу даних, до якого відносяться елементи домену, і довільного логічного виразу, який застосовується до елемента 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 "Прізвища" може бути визначено на базовому типі рядків символів, але в число його значень можуть входити тільки ті рядки, які можуть зображати прізвище (зокрема, такі рядки не можуть починатися з м'якого знака).

В більшості реляційних СУБД поняття домену не використовується, хоча в Oracle V.7 і в InterBase воно вже підтримується.

У мові SQL підтримуються наступні типи даних:

– CHARACTER ( n ) или CHAR ( n ) – символьні рядки постійної довжини в n символів. При завданні даного типу під кожне значення завжди відводиться n символів, і якщо реальне значення займає менш, ніж n символів, то СУБД автоматично доповнює відсутні символи пробілами;

– NUMERIC [(n, m)] - точні числа, тут n - загальна кількість цифр в числі, m - кількість цифр коду точки;

– DECIMAL [(n, m)] - точні числа, тут n - загальна кількість цифр в числі, m - кількість цифр коду точки;

– DEC [(n, m)] - те саме, що і DECIMAL [(n, m)];

– INTEGER або INT - цілі числа;

– SMALLINT - цілі числа меншого діапазону. В більшості реалізацій тип даних INTEGER відповідає цілим числам, збереженим в чотирьох байтах, a SMALLINT - відповідає цілим числам, збереженим в двох байтах. Вибір одного з цих типів визначається розміром числа.

– FLOAT [(n)] - числа великої точності, збережені в формі з плаваючою точкою. Тут n - число байтів, що резервуються під зберігання одного числа. Діапазон чисел визначається конкретною реалізацією;

– REAL - речовинний тип чисел, який відповідає числам з плаваючою точкою, меншою точності, ніж FLOAT;

– DOUBLE PRECISION специфікує тип даних з певною в реалізації точністю більшої, ніж визначена в реалізації точність для REAL.

– VARCHAR (n) - рядки символів змінної довжини;

– NCHAR (N) - рядки локалізованих символів постійної довжини;

– NCHAR VARYING (n) - рядки локалізованих символів змінної довжини;

– BIT (n) - рядок бітів постійної довжини;

– BIT VARYING (n) - рядок бітів змінної довжини;

– DATE - календарна дата;

– ТIМЕSТАМР (точність) - дата і час;

– INTERVAL - часовий інтервал.

Більшість комерційних СУБД підтримують ще додаткові типи даних, які не специфіковані в стандарті. Так, наприклад, практично всі СУБД в тому чи іншому вигляді підтримують тип даних для подання неструктурованого тексту великого обсягу. Цей тип аналогічний типу MEMO в настільних СУБД. Називаються ці типи по-різному, наприклад в Oracle цей тип називається LONG, в DB 2 - LONG VARCHAR, в Sybase і MS SQL Server - TEXT.

Специфіка реалізації окремих типів даних серйозним чином впливає на результати запитів до БД. Особливо це стосується реалізації типів даних DATE і ТIMESТАМР. Тому при перенесенні додатків потрібно бути уважним, на різних платформах вони можуть працювати по-різному, і однією з причин може бути відмінність в інтерпретації типів даних.

Константи дати, часу і часового інтервалу в реляційних СУБД представляються у вигляді строкових констант. Формати цих констант відрізняються в різних СУБД. Крім того, формат представлення дати різний в різних країнах. У більшості СУБД реалізовані способи настройки форматів представлення дат спеціальні функції перетворення форматів дат, як зроблено, наприклад, в СУБД Oracle.

Крім користувальницьких констант в СУБД можуть існувати і спеціальні системні константи. У специфікації SQL :2003 визнані п'ять зумовлених загальних типів, усередині яких можуть бути підтипи (див. таб.2).

Таблиця 2 – Типи даних SQL

|  |  |  |  |
| --- | --- | --- | --- |
| **Тип** | | **SQL реалізація** | |
| Строковий (символьний) | | CHARACTER (чи CHAR) | |
| CHARACTER VARYING (чи VARCHAR) | | | |
| CHARACTER LARGE OBJECT (чи CLOB) | | | |
| Числовий | точні числові типи | | integer;  smallint;  bigint;  numeric;  decimal. |
| приблизні числові типи | | real;  double precision;  float; | |
| логічний (булевий) | | boolean | |
| дати-часу | | date;  time without time zone\*;  time with time zone;  timestamp without time zone;  timestamp with time zone | |

В операторах SQL можуть використовуватися вирази, які будуються за стандартними правилами застосування знаків арифметичних операцій додавання (+), віднімання (-), множення (\*) і ділення (/). Однак в ряді СУБД операція ділення (/) інтерпретується як розподіл без остачі, тому при побудові складних виразів можна отримати результат, який не відповідає традиційній інтерпретації виразу. У стандарт SQL2 включена можливість виконання операцій додавання і віднімання над датами. У більшості СУБД також визначена операція конкатенації (зчеплення) над рядковими даними, позначається вона, на жаль, по-різному. Так, наприклад, для DB 2 операція конкатенації позначається подвійною вертикальною рисою, в MS SQL Server - знаком складання (+).

У стандарті SQL 2 вже введений ряд стандартних вбудованих функцій:

– BIT\_LENGTH (рядок) - кількість бітів в рядку;

– САSТ (значення AS тип даних) - значення, перетворене в заданий тип даних;

– CHAR \_ LENGTH (рядок) - довжина рядка символів;

– CONVERT (рядок USING функція) - рядок, перетворена відповідно до зазначеної функції;

– CURRENT\_DATE - поточна дата;

– CURRENT\_TIME (точність) - місцевий час з вказаною точністю;

– CURRENT \_ TlMESTAMP (точність) - поточні дата і час з вказаною точністю;

– LOWER (рядок) - рядок, перетворена до верхнього регістру;

– OCTED \_ LENGTH (рядок) - число байтів в рядку символів;

– POSITION (перший рядок IN другий рядок) .- позиція, з якої починається входження першого рядка в другу;

– SUBSTRING (рядок FROM n FOR довжина) - частина рядка, що починається з n - го символу і має зазначену довжину;

– TRANSLАТЕ (рядок USING функція) - рядок, перетворена з використанням зазначеної функції;

– TRIM (BOTH символ FROM рядок) - рядок, у якої вилучені всі перші і останні символи;

– TRIM (LEADING символ FROM рядок) - рядок, в якої вилучені всі перші зазначені символи;

– TRIM (TRAILING символ FROM рядок) - рядок, в якої вилучені останні зазначені символи;

– UPPER (рядок) - рядок, перетворена до верхнього регістру.

У стандарті SQL 2 вже введений ряд стандартних вбудованих функцій:

**Рядковий тип даних**

Рядкові дані (послідовності символів) мають три головні строкові типи. Для стовпця таблиці можна вказати тип character (n) або char(рядок фіксованої довжини), де n - максимальна кількість символів, що містяться в рядку. Якщо (n) не вказано, то передбачається, що рядок складається з одного символу. Якщо в стовпець типу character (n) вводиться m < n символів, то позиції, що залишилися, заповнюються пропусками.

Тип даних CHARACTER VARYING (n) або VARCHAR(рядок змінної довжини) застосовується тоді, коли дані, що вводяться, мають різну довжину і небажано доповнювати їх пропусками. При цьому зберігається тільки та кількість символів, яку ввів користувач. В даному випадку вказівка максимальної кількості символів обов'язкова (на відміну від character).

Дані типів CHARACTER і CHARACTER VARYING можуть брати участь в одних і тих же строкових операціях.

Тип даних CHARACTER LARGE OBJECT (CLOB – великий символьний об'єкт) використовується для представлення дуже великих символьних рядків (наприклад, статей, книг і тому подібне). У деяких БД цей тип називається memo, а в інших - text. За даними цього типу можна виконувати не усі операції, передбачені для типів CHARACTER і CHARACTER VARYING. Так, їх не можна використовувати в операціях порівняння, за винятком рівності і нерівності. Крім того, стовпці цього типу не можуть бути первинними і зовнішніми ключами, а також бути оголошені як унікальні значення, що мають. Інакше кажучи, при створенні таблиць за допомогою оператора CREATE і оголошенні стовпців типу CLOB не можна використовувати ключові слова PRIMARY KEY, FOREIGN KEY і UNIQUE.

**Числовий тип даних**

Числовий тип даних може бути двох видів точний і приблизний. Точні числові типи дозволяють точно виразити значення числа. Деякі величини мають дуже великий діапазон значень, і в таких випадках досить обмежитися деяким наближеним їх представленням з урахуванням технічних можливостей комп'ютера (розмірів регістра).

До ***точних числових*** відносяться наступні п'ять типів: integer, smallint, bigint, numeric та decimal.

*Integer* - ціле (без дробової частини) число. Кількість розрядів (точність) залежить від реалізації SQL. У деяких реалізаціях числа цього типу лежать в діапазоні від - 2 147 483 648 до 2 147 483 647 (четырехбайтное ціле число).

*Smallint* - мале ціле число. Кількість розрядів залежить від реалізації SQL, але не більше кількості розрядів integer в цій же реалізації. У деяких реалізаціях числа цього типу лежать в діапазоні від - 32 768 до 32 767 (двобайтове ціле число).

*Bigint* - велике ціле число. Кількість розрядів залежить від реалізації SQL і перевищує кількість розрядів числа типу integer.

*Numeric (x, у)* - число, в якому усього х розрядів (точність), з яких у розрядів (масштаб) відводиться для дробової частини. Якщо у не вказано (numeric (х)), то для дробової частини відводиться кількість розрядів, встановлена в системі за умовчанням. Якщо не вказані ні х, ні у (numeric), тo приймаються обоє ці величини, встановлені за умовчанням. Наприклад, якщо вказаний тип numerc (6, 2), то максимальне значення числа дорівнює 9999.99.

*Decimal (x, у)* - десяткове число, в якому усього х розрядів, з яких у розрядів відводяться для дробової частини. Якщо х або/і у не вказані, то набувають значень за умовчанням. Цей тип дуже схожий на numeric.

Відмінність полягає в тому, що якщо в decimal (х, у) вказані х і у менше, ніж допустимі реалізацією SQL, то використовуватимуться останні. Якщо х і у не вказані, то застосовується система умовчань. Наприклад, ви задали для стовпця тип decimal (6, 2). Якщо реалізація SQL дозволяє, то в цей стовпець можна ввести числа, що перевищують 9999.99. На відміну від decimal (х, у), тип numeric (x, у) жорстко задає діапазон можливих значень числової величини.

До ***приблизних числових*** типів відносяться наступні три типи: real, double precision та Float.

*Real* - дійсне число одинарної точності з плаваючою розділовою

точкою (ця точка "плаває", з'являючись в різних місцях числа). Наприклад, 5.25, 5.257, 5.2573. Точність представлення числа залежить від реалізації SQL і устаткування. Наприклад, 32-бітовий комп'ютер дає велику точність, чим 16-бітовий;

*Double precision* - дійсне число подвійної точності з плаваючою розділовою точкою. Точність представлення числа залежить від реалізації SQL і устаткування. Застосовується для представлення наукових даних (наприклад, результатів вимірів) в широкому діапазоні значень, тобто як дуже малих (близьких до 0), так і дуже великих;

*Float (x)* - дійсне число з плаваючою розділовою точкою і мінімальною точністю х, що займає не більше 8 байтів. Якщо комп'ютер може підтримати вказану точність, використовуючи апаратну одинарну точність, то система використовуватиме арифметику одинарної точності. Якщо вказана точність вимагає арифметики з подвійною точністю, то система використовуватиме її.

Цей тип слід застосовувати, якщо передбачається можливість перенесення бази даних на іншу апаратну платформу, що відрізняється розмірами регістрів. Приклад значення типу float : 5.318E-24 (тобто 5.318, помножене на 10 в мірі - 24). Таку ж форму представлення мають і числа типу real і double precision.

При створенні таблиць цілочисельні типи застосовуються для стовпців, що містять різного роду ідентифікатори, наприклад, номера (коди) клієнтів, товарів, замовлень і тому подібне. Зрозуміло, чи її вміст стовпця має бути цілим числом (наприклад, кількість ящиків, пляшок, штук і тому подібне), то тип цього стовпця природно визначити як integer, smallint або bigint.

**Логічні дані**

У SQL тип даних boolean (булевий) має три значення - true, false і unknown. Значення unknown (невідоме) було введене для позначення результату, що виходить при порівнянні зі значенням null (невизначене).

Якщо користувач ще не ввів в елемент таблиці ніякого значення, то цей "порожній" осередок містить значення null, що інтерпретується як невідоме або невизначене значення.

Результатом будь-якої операції порівняння true або false з null або з unknown завжди являється unknown.

У SQL -виразах логічні значення полягають в лапки, наприклад, 'TRUE' або 'true'.

**Дата і час**

Тип data (дата) призначений для зберігання значень дати, елементи яких розташовані в наступному порядку: рік (4 цифри), дефіс (-), місяць (2 цифри), дефіс, день (2 цифри).

Таким чином, значення дати займають 10 позицій, наприклад, 2005-10-02.

Дані цього типу можуть містити будь-яку дату з 0001 року по 9999 рік.

Для представлення часу передбачено два типи: time without time zone та time with time zone.

Time without time zone (час без часового поясу) призначена для зберігання значень часу, елементи яких розташовані в наступному порядку :годинник, двокрапка, хвилини, двокрапка, секунди. Годинник і хвилини представляються двома цифрами, а секунди можуть бути представлені двома і бо¬леї цифрами (якщо вимагається дробова частина), наприклад 18:35:19.547. Довжина дробової частини секунд залежить від реалізації, але внутрішнє представлення часу повинне мати не менше 6 цифр. За замовчанням час цього типу представляється без дробової частини секунд.

Time with time zone (час з часовим поясом) - такий же тип даних, як і time without time zone. Відмінність полягає лише в тому, що до значення часу додається ще і інформація про різницю між місцевим і всесвітнім часом. Всесвітній час (Universal Time Coordinated, UTC) – це час по Грінвічу, тобто час нульового меридіана, що ходить через м.Грінвіч у Великобританії (Greenwich Mean Time, GMT). Значення різниці між локальним і всесвітнім часом знаходиться в діапазоні від -12:59 до 13:00.

Довжина даних даного типу дорівнює довжині даних типу time without time zone плюс 6, оскільки додаткова інформація про різницю часів займає 6 позицій (дефіс, знак (+) або (-), 2 цифри для годинника, двійкова, 2 цифри для хвилин).

Для одночасного представлення дати і часу служать наступні два типи: timestamp without time zone та timestamp with time zone.

Timestamp without time zone (дата і час без часового поясу). Елементи даних цього типу мають такі ж характеристики, як і для даних типу date і time without time zone, за винятком одного: дані типу timestamp without time zone пo умовчанню мають 6 цифр в дробовій частині секунд, а не 0, як в типі time without time zone. Для вказівки кількості цифр в дробовій частині використовується синтаксис timestamp without time zone (л). Якщо дробової частини немає, то дані займають 19 позицій: 10 позицій для дати, один пропуск і 8 позицій для часу. Якщо визначена дробова частина, то довжина даних дорівнює 20 плюс кількість цифр в дробовій частині секунд;

Timestamp with time zone (дата і час з часовим поясом) - такий же тип даних, як і timestamp with time zone. Відмінність полягає в тому, що до значення часу додається ще і інформація про різницю між місцевим і всесвітнім часом. Додаткова інформація займає 6 позицій. Дані типу timestamp with time zone без дробової частини займають 25 позицій, з дробовою частиною - 26 плюс кількість цифр в дробовій частині секунд.

Щоб представити в SQL - виразі дату, час або дату-час, необхідно використовувати функцію cast об приведення до заданого типу. Допустимо, в таблиці продажу є стовпець дата типу data.

Щоб отримати відомості з цієї таблиці за період після 2005-09-30, слід виконати такий запит:

SELECT \* FROM Sales WHERE Data > CAST ('2005-09-30' AS DATE);

Тут рядок, що містить дату, приводиться до типу data, результат бере участь в операції порівняння з даними стовпця Data.

У мові SQL є три функції, які ***повертають поточну дату*** і час :

- current\_date - повертає поточну дату (тип date). Наприклад, 2005-06-18;

- current\_time (число) - повертає поточний час (тип time).

Цілочисельний параметр число вказує точність представлення секунд. Наприклад, при число = 2 секунди будь представлені з точністю до сотих (дві цифри в дробній частині) : 12:39:45.27;

- current\_timestamp (число) - повертає дату і час (тип TIMESTAMP), наприклад 2005-06-18 12:39:45.27. Цілочисельний параметр число вказує точність представлення секунд.

**Структура мови SQL**

Мова SQL має два основних компоненти (див.рис.1):

– мову DDL (Data Definition Language), призначену для визначення структур бази даних та управління доступом до даних (оператори CREATE TABLE; DROP TABLE; ALTER TABLE; CREATE INDEX; DROP INDEX);

– мову DML (Data Manipulation Language), призначену для вибірки і оновлення даних ( оператори SELECT, INSERT, UPDATE, DELETE, COMMIT – фіксація змін, ROLLBACK – відміна внесених змін) (рис. 2).

Після створення загальної структури бази даних можна приступити до створення таблиць, якими є стосунки, що входять до складу проекту бази даних.

Базовий синтаксис оператора створення таблиці має наступний вигляд:

CREATE TABLE ім’я\_таблиці (ім’я\_стовбця тип\_даних [NULL | NOT NULL ] [,...n])

Головне в команді створення таблиці – визначення імені таблиці і опис набору імен полів, які вказуються у відповідному порядку. Крім того, цією командою обмовляються типи даних і розміри таблиці

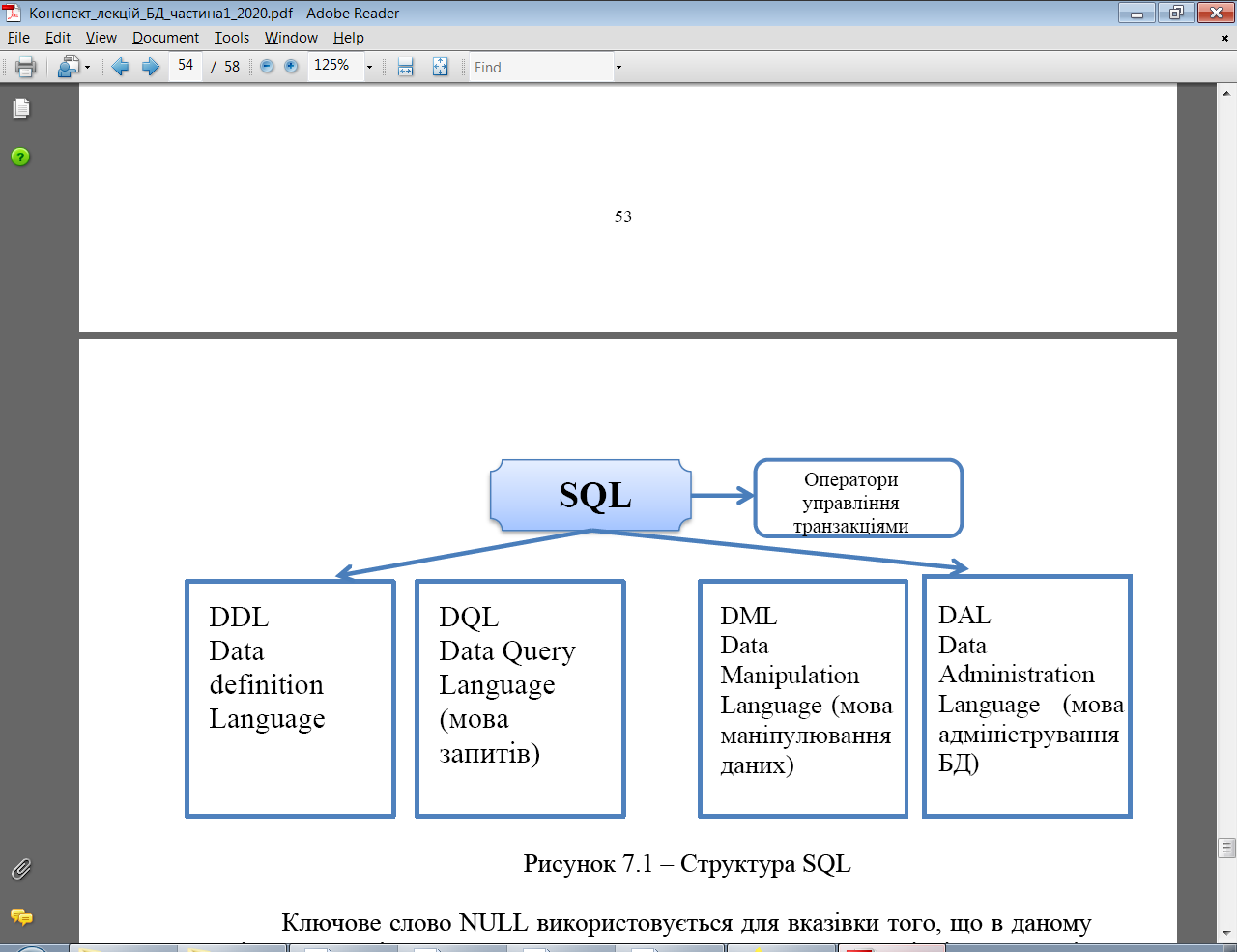


Рисунок 1 – Структура SQL

Ключове слово NULL використовується для вказівки того, що в даному стовпці можуть міститися значення NULL. Значення NULL відрізняється від пропуску або нуля – до нього удаються, коли необхідно вказати, що дані недоступні, опущені або недопустимі. Якщо вказано ключове слово NOT NULL, то будуть відхилені будь-які спроби помістити значення NULL в даний стовпець. Якщо вказаний параметр NULL, приміщення значень NULL в стовпець дозволене. За умовчанням стандарт SQL передбачає наявність ключового слова NULL.

Ми використовували спрощену версію оператора CREATE TABLE стандарту SQL. Його повна версія наводиться при обговоренні питань забезпечення цілісності даних.

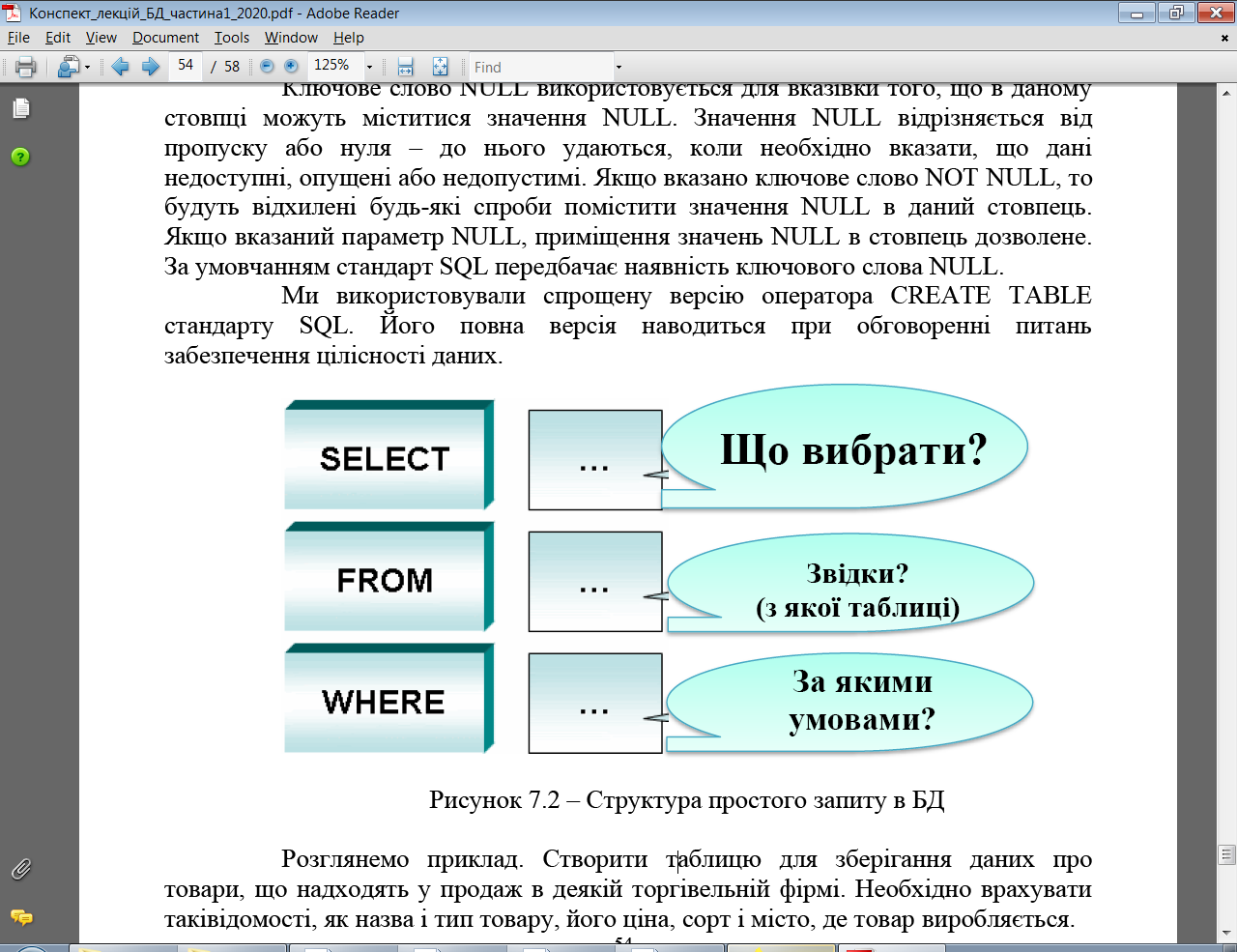


Рисунок 2

Приклад. Створити таблицю для зберігання даних про товари, що надходять у продаж в деякій торгівельній фірмі. Необхідно врахувати такі відомості, як назва і тип товару, його ціна, сорт і місто, де товар виробляється.

Реалізація приведена нижче.

CREATE TABLE Товар

(Назва VARCHAR(50) NOT NULL

Ціна MONEY NOT NULL

Тип VARCHAR(50) NOT NULL

Сорт VARCHAR(50)

МістоТовару VARCHAR(50))

**Транзакція БД**

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. 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 сериалізації можливі ситуації конфліктів між двома або більше транзакціями з доступу до об'єктів БД. В цьому випадку для підтримки сериалізації необхідно виконати відкат (ліквідувати всі зміни, вироблені в БД) однієї або більше транзакцій.

Отже, SQL можна в повній мірі віднести до традиційних мов програмування, він не містить традиційні оператори, що керують ходом виконання

56

програми, оператори опису типів і багато іншого, він містить тільки набір стандартних операторів доступу до даних, що зберігаються в базі даних. Оператори SQL вбудовуються в базова мова програмування, яким може бути будь-який стандартний мову типу C ++, Pascal, COBOL і т. Д. Крім того, оператори SQL можуть виконуватися безпосередньо в інтерактивному режимі.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Стандарт вищої освіти України: перший (бакалаврський) рівень, галузь знань 12 − Інформаційні технології, спеціальність 126 − Інформаційні системи та технології. Затверджено Наказом Міністерства освіти і науки України 12.12.2018 р. № 1380. – 17 с.

2. Ben Forta. SQL in 10 Minutes a Day, Sams Teach Yourself: Sams Publishing; 5th edition. – 18 Aug. 2020. – 256p. ISBN-10 : 0135182794, ISBN-13 : 978-0135182796.

3. Гайдаржи В., Ізварін І.Бази даних в інформаційних системах: Навчальний посібник. – Тернопіль: Навчальна книга.– 2018.– 418 с.

4. Jamie Chan. SQL: Learn SQL (using MySQL) in One Day and Learn It Well. SQL for Beginners with Hands-on Project. – 2018. – 166p. ASIN : B07K374J19.

5. Мулеса О.Ю. Інформаційні системи та реляційні бази даних. Навч.посібник. – Електронне видання, 2018. – 118 с.

6. Корнієнко С.К. Проектування інформаційного забезпечення автоматизованих систем. Навч. Посібник. – Запоріжжя: ЗНТУ, 2015. – 210 с.

7. Карпуша В.Д., Панченко Б.Є. Моделювання та проектування реляційних баз даних : навч. посіб. – Суми : СДУ, 2010.

8. Сенів М. М., Яковина В. С. Безпека програм та даних : навч. посіб. Львів : Видавництво Львівської політехніки, 2015.

9. Завадський І.О. Основи баз даних: [Навч. посіб.] / І.О. Завадський. – К. : Видавець І.О. Завадський, 2011. – 192 с.

10. Балик Н., Мандзюк В.Бази даних MySQL: Навчальний посібник. – Тернопіль: Навчальна книга – Богдан, 2010.– 160 с.

11. Гайдаржи В.І., Дацюк О.А. Основи проектування та використання баз даних : Навч. посібник Київ : Політехніка, 2004. – 166c.

12. Адміністрування баз даними // Режим доступу: [http://firebirdsql.org/manual/ru/migration- mssql-db-admin-ru.html](http://firebirdsql.org/manual/ru/migration-%20mssql-db-admin-ru.html).

13. Управління і адміністрування баз даними // Режим доступу: http://www.interface.ru/home.asp?artId=50&cId=3.