Тема № 4

Створення реляційних таблиць

Мета: Навчитися створювати логічну структуру бази даних, використовуючи засоби мови SQL. Засвоїти синтаксис інструкції SQL CREATE TABLE.

Обладнання, матеріали: встановлений та налаштований відповідно до вимог, вказаних в завданні до Теми № 1 локальний сервер MySQL.

Теоретичні відомості:

Створення таблиць бази даних

Реляційні таблиці відрізняються від довільних двомірних таблиць за такими ознаками:

- рядки не нумеруються, порядок їх слідування не має значення;
- стовпчики не нумеруються, однозначно визначаються іменами, в межах однієї таблиці всі імена стовпчиків – унікальні;
- на перетині будь-якого стовпчика і будь-якого рядка знаходиться атомарне (неподільне) значення;
- кожна таблиця має **первинний ключ** – набір полів (від одного до всіх полів), таких, що задаючи значення цьому набору полів можна повністю ідентифікувати весь запис, і притому єдиний, тобто в будь-яких двох записів реляційної таблиці значення первинного ключа не можуть збігатися.

Створення таблиць в будь-якій *реляційній* СУБД базується на застосуванні мови визначення даних SQL DDL (Data Definition Language), в якій є конструкції для створення і видалення таблиць, визначення обмежень на значення полів таблиць і т.д.

Варто відзначити, що існує єдиний стандарт мови SQL (ANSI SQL92, SQL3 та ін.), проте кожна СУБД розширює цей стандарт по-своєму, тому, знаючи стандартний SQL, можна працювати з будь-якою реляційною СУБД, але при роботі з конкретною СУБД слід з'ясувати особливості використовуваного діалекту SQL.

У даному посібнику всі інструкції SQL подаються для СУБД MySQL.

Інструкція SQL на створення таблиці виглядає так:

```
CREATE  [TEMPORARY] TABLE [IF NOT EXISTS] ім'я_таблиці (
        визначення_поля[, визначення_поля,...n]
        обмеження_на_таблицю[,...n]     )
        [ENGINE [=] тип_таблиці]
        [CHARACTER SET [=] кодування]
;
```

де:

ім’я_поля, ім’я_таблиці – припустимі імена в MySQL.

Припустимими є імена довжиною до 64 символів, що не містять крапки «.», знаку оклику «!», символу зворотних лапок «`» та символів квадратних дужок «[ ]», і починаються не з пропуску.

Назви таблиць та полів, що включають пропуски, а також символи національних кодувань, вміщуються у *зворотні лапки*, наприклад `order details`. Те ж саме потрібно буде робити у всіх посиланнях на таблицю.

Параметр TEMPORARY задає створення тимчасової таблиці, існуючої протягом поточного сеансу. По завершенні сеансу таблиця видаляється. Тимчасовим таблицям можна привласнювати імена інших таблиць, роблячи останні тимчасово недоступними.

Параметр IF NOT EXIST пригнічує виведення повідомлень про помилку в разі, якщо таблиця з вказаним ім’ям вже існує.

Перед ім’ям таблиці можна вказувати ім’я бази даних, відокремлене крапкою, наприклад, northwind.orders. Якщо це не зроблено, таблиця буде створена у базі даних, яка встановлена за умовчанням.

Дозволяється створювати таблиці без стовпців, проте у більшості випадків специфікація хоча б одного стовпця все ж таки присутня.

визначення_поля має вигляд

```
ім’я_поля тип [(розмір)] [обмеження_на_поле]
```

тип – див. таблицю типів даних MySQL (Табл. 6.1). Специфікація типу включає назву типу та його розмірність. За умовчанням стовпці приймають значення NULL. Параметр NOT NULL забороняє подібну поведінку.

У будь-якого стовпця є *значення за умовчанням*. Якщо воно не зазначене, MySQL вибере його самостійно. Для стовпців, що приймають значення NULL, значенням за умовчанням буде NULL, для рядкових стовпців - порожній рядок, для чисельних стовпців - нуль. Змінити цю установку дозволяє пропозиція DEFAULT.

Поля-лічильники, створювані за допомогою прапора AUTO_INCREMENT, ігнорують значення за умовчанням, так як у них записуються порядкові номери. Тип лічильника повинен бути беззнаковим цілим. У таблиці може бути присутнє лише одне поле-лічильник. Таким полем не обов'язково буде первинний ключ.

обмеження_на_поле має вигляд:

```
[CONSTRAINT ім’я_обмеження] {PRIMARY KEY | UNIQUE | NOT NULL}
```

обмеження_на_таблицю має вигляд:

```
[CONSTRAINT ім’я_обмеження]
 {PRIMARY KEY (ключове_поле1[, ключове_поле2 [, ...]])      |
 UNIQUE (унікальне_поле1[, унікальне_поле2 [, ...]])        |
 NOT NULL (непорожнє_поле1[, непорожнє_поле2 [, ...]])      |
```

FOREIGN KEY (поле_посилання1[, поле_посилання2 [, ...]])
REFERENCES батьківськаТаблиця [(ключове_поле1 [, ключове_поле2 [, ...]] [ON DELETE опція] [ON UPDATE опція])]}

У стандарті SQL речення CONSTRAINT використовується для налаштування обмежень на значення полів запису (зокрема, унікальність – UNIQUE, обов'язковість – NOT NULL), для встановлення відношень між таблицями (FOREIGN KEY…REFERENCES…), а також для налаштування обмежень на таблицю в цілому.

Відзначимо, що *імена обмежень* у виразах CONSTRAINT в межах однієї бази даних *мають бути різними*.

Інструкція ENGINE притаманна діалекту SQL MySQL. Вона відповідає за тип таблиці, з огляду на спосіб збереження та обробки даних. Найчастіше використовуються два типи таблиць – InnoDB та MyISAM. Їх особливості подані у Додатку Б.

Тип таблиць InnoDB підходить у випадках, коли потрібно контролювати дії над даними, та мати можливість відмінити деякі дії. Такий тип таблиць підтримує транзакції (див. Тему № 14). Тип таблиць MyISAM часто застосовують, коли дані змінюються дуже рідко, і менше вимог до контролю змін. З усіма типами таблиць та їх особливостями можна ознайомитися у [2] переліку додаткової літератури.

Інструкція CHARACTER SET примушує встановити на окрему таблицю кодування символів, відмінне від встановленого для усієї бази даних.

Видалення таблиць

Для того, щоб видалити таблицю, спочатку треба переконатися в тому, що вона існує. Це можна перевірити за допомогою команди SHOW TABLES.

Для видалення таблиці використовується оператор DROP TABLE.

Оператор DROP TABLE має наступний синтаксис:

DROP TABLE [IF EXISTS] таблиця;

Параметр IF EXISTS пригнічує виведення повідомлення про помилку, що видається у разі, якщо задана таблиця не існує. Можна зазначати декілька назв таблиць, розділяючи їх комами.

Батьківські та дочірні таблиці

Розглянемо на прикладі, як використовуються речення FOREIGN KEY...REFERENCES… оператору SQL CREATE TABLE. Нехай перед нами стає задача розробити систему таблиць для бази даних weather_archive.

Спочатку створимо таблицю wind_direction, в якій будуть зберігатися скорочені («clm», «n», «ne», «nne») та повні назви напрямів вітру («calm», «north», «north-east», «north-north-east» та інші).

```
CREATE TABLE wind_direction (
    wind_type_id int AUTO_INCREMENT NOT NULL,
    wind_direction_short char(3) NOT NULL,
    wind_direction_title varchar(15),
    CONSTRAINT wd1 PRIMARY KEY (wind_type_id)
    );
```

Створюється таблиця з трьох полів, причому поле wind_type_id стає первинним ключем, і для цього поля за умовчанням створюється індекс. Для визначення первинного ключа тут використовується обмеження на поле з ім'ям wd1.

Тепер створимо таблицю weather_archive, яка міститиме значення швидкості (у метрах за секунду) та напряму вітру за вимірами, які проводилися чотири рази на день.

```
CREATE TABLE weather_archive (
    measurement_id int AUTO_INCREMENT NOT NULL,
    measurement_date date NOT NULL,
    measurement_time time NOT NULL,
    wind_type_id int NOT NULL,
    wind_speed smallint NOT NULL,
    CONSTRAINT wa1 PRIMARY KEY (measurement_id),
    CONSTRAINT wa2 FOREIGN KEY (wind_type_id)
        REFERENCES wind_direction (wind_type_id));
```

Створюється таблиця, в якій первинний ключ – поле лічильника measurement_id, а значення поля wind_type_id обмежуються значеннями поля wind_type_id таблиці wind_direction. При створенні цієї таблиці автоматично створюються і два індекси: індекс за первинним ключем та індекс за зовнішнім ключем. На Рис. 6.1 (зробленому за допомогою інструменту Reverse Engineering з пакету MySQL Workbench) показано відповідну реляційну схему даних.

Як видно, між цими таблицями встановлено зв'язок **«один до багатьох»**, який з'явився завдяки визначенню **зовнішнього ключа** FOREIGN KEY … REFERENCES … При цьому таблиця wind_direction є **батьківською**, а weather_archive – **дочірньою**.

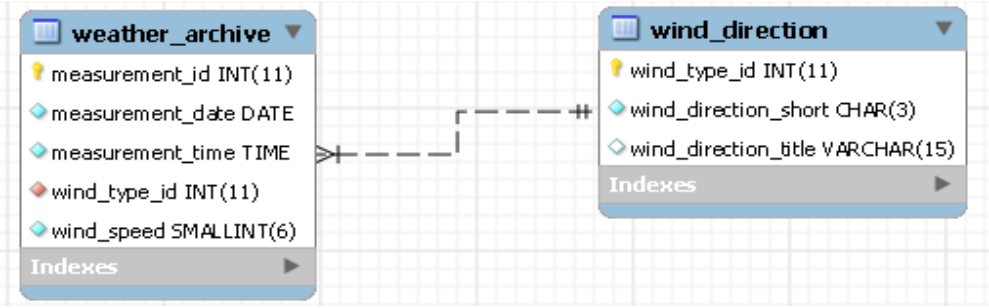

Рисунок 6.1 – wind_direction – батьківська, weather_archive – дочірня таблиці.

Визначення **зовнішнього ключа для поля(-ів)** в дочірній таблиці звужує область можливих значень цього стовпчика(-ів) такими, які вже існують в батьківській таблиці. Таким чином, досягається *цілісність даних*. **Цілісність даних** означає систему правил, використовуваних в СУБД для підтримки зв'язків між записами в зв'язаних таблицях, а також для забезпечення захисту від випадкового видалення або зміни пов'язаних даних.

Встановити *цілісність даних* між таблицями можна за умов виконання наступних правил:

- типи даних для пов'язаних полів дочірньої та батьківської таблиці мають збігатися;
- принаймні в батьківській таблиці поле, обране для зв'язку, має бути обов'язковим і унікальним. Як правило, це поле первинного ключа. При цьому між таблицями встановлюється зв'язок **«один до багатьох»**;
- якщо і в батьківській, і в дочірній таблиці пов'язані поля є обов'язковими та унікальними, то встановлюється зв'язок **«один до одного»**.

При визначенні зовнішніх ключів дуже зручно встановити поведінку СУБД при видаленні або поновленні рядку у батьківській таблиці.

Дійсно, існує два варіанта поведінки СУБД в таких випадках. Перший полягає у **забороні зміни /видалення** рядку батьківської таблиці, якщо на цей рядок є посилання з будь-якої дочірньої таблиці (дія RESTRICT). Другий – у **каскадній зміні/видаленні** пов'язаних рядків у дочірніх таблицях (дія CASCADE).

Для того, щоб вказати потрібну поведінку, при створенні **дочірньої** таблиці оператором CREATE TABLE речення на створення зовнішнього ключа буде таким:

```
[CONSTRAINT ім'я_обмеження]
  FOREIGN KEY (поле_посилання1[, поле_посилання2 [, ...]])
       REFERENCES батьківськаТаблиця [(ключове_поле1[,
                    ключове_поле2 [, ...]]
       ON DELETE прапор
       ON UPDATE прапор
              )]
```

де:

прапор = {CASCADE | RESTRICT | SET NULL}

Зазвичай СУБД пропонують кілька способів створення таблиць.

Перший спосіб полягає в тому, що будь-яку таблицю можна створити, написавши відповідну інструкцію SQL CREATE TABLE.

Другий спосіб створення таблиць полягає у використанні візуальних засобів. Основна ідея подібних конструкторів полягає в тому, щоб легко створити список полів конкретної таблиці та настроїти тип даних та властивості цих полів.

Для MySQL зручно застосовувати інструмент MySQL Workbench, який має візуальні засоби для створення та пов'язування батьківських й дочірніх таблиць. У режимі конструктора (зовнішній вигляд див. на Рис. 6.2) кожен стовпчик таблиці представлено окремим рядком. Для визначення стовпчика слід вказати його ім'я, тип даних, і настроїти додаткові властивості полів.

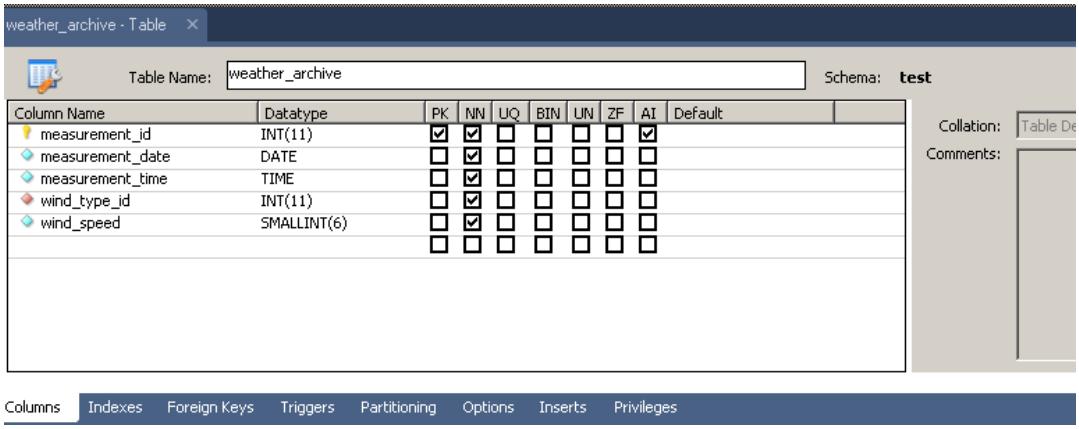

Рисунок 2.3 – Зовнішній вигляд конструктора таблиць.

Звернемо увагу на додаткові властивості, доступні на вкладці **Columns**: PK – є полем первинного ключа, NN – NOT NULL, UQ – UNIQUE, BIN – поле є бінарним, ZF – якщо значення поля не вказано, занести значення 0, AI – AUTO_INCREMENT, Default – значення за умовчанням.

На вкладці **Indexes** перераховані індекси, які або створені MySQL за умовчанням (для первинного та зовнішніх ключів), або окремо, розробником структури бази даних.

На вкладці **Foreign Keys** можна продивитися зовнішні ключі для таблиці.

Вкладка **Options** містить параметри збереження даних, включаючи шаг для поля лічильника.

Детальний опис типів даних, які підтримуються у MySQL, можна отримати у [3](з переліку додаткової літератури). Коротко перелічимо їх групи та назви (див. Табл. 6.1).

Таблиця 6.1 – Типи даних в MySQL.

Тип даних	**Назва типу у MySQL**	**Опис типу даних**

Тип даних	Назва типу у MySQL	Опис типу даних
Текстовий	[NATIONAL] CHAR (M), NCHAR(M) [BINARY]	Текст довжиною від 0 до 255 символів.
	[NATIONAL] VARCHAR (M) [BINARY], NVARCHAR(M) [BINARY]	
	Вказання прапору NATIONAL означає, що у полі зберігатимуться текстові значення у національних (двобайтних) кодуваннях. Прапор чутливості до регістру, BINARY, якщо вказаний, примушує СУБД виконувати сортування та порівняння значень поля з урахуванням регістру (див. також Тему № 2).	
Цілі числа	TINYINT [(M)] [UNSIGNED] [ZEROFILL]	Від -128 до +127 (від 0 до 255)
	SMALLINT [(M)] [UNSIGNED] [ZEROFILL]	Від -2^15 до +2^15-1 (від 0 до 2^16-1)
	MEDIUMINT [(M)] [UNSIGNED] [ZEROFILL]	Від -2^23 до +2^23-1 (від 0 до 2^24-1)
	INT [(M)] [UNSIGNED] [ZEROFILL] або INTEGER [(M)] [UNSIGNED] [ZEROFILL]	Від -2^31 до +2^31-1 (від 0 до 2^32-1)
	BIGINT [(M)] [UNSIGNED] [ZEROFILL]	Від -2^63 до +2^63-1 (від 0 до 2^64-1)
	Прапор UNSIGNED вказує на невід'ємність значень поля. Прапор ZEROFILL вказує СУБД на те, що пусті поля треба заповнити нулями.	
Дійсні числа	FLOAT(точність)	Число з плаваючою комою. точність не перевищує 24 знака.
	FLOAT [(M,D)]	Число з плаваючою комою звичайної точності. Атрибут M вказує кількість знаків для користувача, атрибут D - кількість розрядів після коми.
	DOUBLE [(M,D)] DOUBLE PRECISION [(M,D)]	Число з плаваючою комою подвійної точності. Атрибут M вказує кількість знаків для користувача, атрибут D - кількість розрядів після коми.

Тип даних	Назва типу у MySQL	Опис типу даних
	DECIMAL[(M[,D])], DEC [(M[,D])], NUMERIC [(M[,D])]	«Неупаковане» число з плаваючою комою. Число зберігається у вигляді строки, при цьому кожній десятковій цифрі відповідає один символ. Розділовий знак коми, а також знак «-» для від'ємних чисел не враховуються в M.
Логічний	BIT, BOOL (еквівалентні TINYINT(1))	Логічні значення.
Дата/Час	DATE	Дата, від '1000-01-01' до '9999-12-31'. Формат збереження 'YYYY-MM-DD'
	DATETIME	Комбінація дати та часу, від '1000-01-01 00:00:00' до '9999-12-31 23:59:59'. Формат збереження 'YYYY-MM-DD HH:MM:SS'
	TIME	Час, від '-838:59:59' до '838:59:59'. Формат збереження 'HH:MM:SS'
	TIMESTAMP [(M)]	Часова мітка, від '1970-01-01 00:00:00' до деякого значення часу у 2037 році. Формат виводу для користувача залежить від M. M=14: 'YYYYMMDDHHMMSS' M=12: 'YYMMDDHHMMSS' M=8: 'YYYYMMDD' M=6: 'YYMMDD'
	YEAR [(2 | 4)]	Рік, або від 1901 по 2155, або від 1970 до 2069
Великий текст або бінарний об'єкт	TINYTEXT, TINYBLOB	Поле BLOB / TEXT з максимальною довжиною 255 символів
	TEXT, BLOB	Поле BLOB / TEXT з максимальною довжиною 2^16-1 символів
	MEDIUMTEXT, MEDIUMBLOB	Поле BLOB / TEXT з максимальною довжиною 2^24-1 символів
	LONGTEXT, LONGBLOB	Поле BLOB / TEXT з максимальною довжиною 2^32-1 символів
Пере-числення	ENUM ('знач-1', ..., 'знач-n')	Поле може мати лише одне з переліку значень.
Набір	SET ('знач-1', ..., 'знач-n')	Поле може мати нуль або більше значень, кожне з яких обрано з переліку значень

Зміна структури таблиці

Іноді виникають ситуації, коли потрібно змінити структуру вже існуючої таблиці – перейменувати, додати або видалити поле, змінити тип даних поля тощо. Найпростіше це зробити, якщо таблиця не містить записів, але деякі дії можливо виконати і при наявності записів. Для зміни структури таблиці використовується оператор ALTER TABLE. Повністю формат оператора наведено у [4] (з переліку додаткової літератури). Найпростіші приклади включають перейменування таблиці, поля, зміна типу даних, видалення та додавання поля.

Перейменування таблиці `order details` на `order_details` бази даних northwind виглядатиме як

```
ALTER TABLE  `order details` RENAME `order_details`;
```

Перейменування поля wind_direction_title на wind_direction_long:

```
ALTER TABLE  wind_direction CHANGE wind_direction_title
                wind_direction_long varchar(15);
```

Зміна типу даних поля wind_direction_short з int на enum:

```
ALTER TABLE wind_direction MODIFY wind_direction_short enum (‘clm’, ‘n’, ‘ne’, ‘nne’, ‘e’,..., ‘ssw’);
```

Видалення поля з таблиці

```
ALTER TABLE DROP COLUMN ‘wind_direction_long’;
```

Додавання нового поля до таблиці

```
ALTER TABLE ADD  ‘wind_direction_long’ varchar(15);
```

Завдання:

1. Відкрийте базу даних, створену на локальному екземплярі сервера MySQL за варіантом у завданні до Теми № 2.
2. Створіть за допомогою інструкції CREATE TABLE всі таблиці за варіантом. Проконтролюйте створення первинних та зовнішніх ключів.
3. Створіть схему даних за допомогою MySQL Workbench (команда меню **Database / Reverse Engineer…**). Зробіть екранну копію схеми даних.
4. Використовуючи інструкцію ALTER TABLE та тип даних ENUM, настройте вибір значень для полів з фіксованих списків значень, за варіантом.
5. Занесіть декілька значень до кожної з таблиць.

Варіанти:

Підкресленням позначено ключові поля.

1. БД «Кінофільми» (взято з [6] переліку додаткової літератури)
   Кінозірки (ім'я, стать, дата_народження, адреса, фото)
   Кінофільми (назва, рік_виходу, тривалість, ознака_кольору, назва_студії, ім'я_продюсера), де значення поля «ознака_кольору» обирається зі списку «кольоровий, чорно-білий».
   Студії (назва_студії, адреса, телефон, ім'я_президента, web-сайт)
   ЗіркиФільми (назва_фільму, рік_виходу, ім'я_кінозірки)

2. БД «Книговидання»
   Автори (код_авт, ім'я, стать, дата_народження, рік_смерті, адреса, телефон, фото, email)
   Книги (код_кн, ISBN, назва, рік_публікації, сторінки, назва_видавництва, вид_обкладинки, тип_обкладинки).
   Видавництва (назва_видавництва, логотип, адреса, телефон, ім'я_контактної_особи)
   АвториКниги (код_кн, код_авт)

3. БД «Змагання по плаванню»
   Спортсмени (код, ім'я, стать, рік_народження, країна, місто, фото)
   Дистанції (код_дистанції, довжина, стиль), де значення поля «стиль» обираються зі списку «вільний, вільний на спині, брас, брас на спині, батерфляй, естафета, комплекс»
   ЕтапиЗмагання (номер_етапу, опис), де значення поля «опис» обираються зі списку «1/8 фіналу, 1/4 фіналу, 1/2 фіналу, фінал».
   Запливи (код, код_дистанції, номер_етапу, результат, є_рекорд)

4. БД «Змагання по фігурному катанню»

Спортсмени (код, ім’я, стать, рік_народження, країна, місто, тренер, звання, фото, категорія), де значення поля «категорія» обираються зі списку «одиночне катання, парне катання, танці на льоду»
ЕтапиЗмагання (номер_етапу, опис), де значення поля «опис» обираються зі списку «1/8 фіналу, 1/4 фіналу, 1/2 фіналу, фінал».
Виступи (код_виступу, номер_етапу, вид_програми, код, код_партнера, музика), де значення поля «вид_програми» обираються зі списку «коротка, обов’язкова, вільна»
РезультатиВиступу (код_виступу, країна_судді, оцінка_арт, оцінка_техн)

5. БД «Футбольні клуби» (адапт. з [6] переліку додаткової літератури)
Гравці (код_гравця, ім’я, дата_народження, місто_народження, фото)
Команди (код_команди, назва_команди, місто, футбольний_клуб)
Тренери (код_тренера, ім’я, дата_народження, місто_народження, фото)
Сезони (код_гравця, рік, номер_гравця_у_команді, код_команди, забито_м’ячів, голевих_ситуацій, код_тренера)

6. БД «Чемпіонат світу по футболу. Основний турнір»
Команди (назва_команди, країна)
Гравці (код_гравця, ім’я, спеціалізація, номер_в_команді, назва_команди)
Игри (номер_гри, назва_команди1, назва_команди2, результат1, результат2, додатковий_час, с), де значення поля «специализация» обирається зі списку «1/16 фіналу, 1/8 фіналу, ¼ фіналу, ½ фіналу, фінал»
Подробиці (номер_гри, час, подія, код_гравця1, код_гравця2) де значення поля «подія» обирається зі списку «гол, заміна, жовта картка, красна картка»

7. БД «Автомобілі»
Автомобілі (марка, модель, маса, макс_швидкість, витрата_палива, заводська_ціна, фото, код_кузова, код_двигуна, виробник)
Кузови (код_кузова, назва, кількість_місць, кількість_дверей)
Двигуни (код_двигуна, тип_палива, об’єм, потужність), де значення поля «тип_палива» обирається зі списку «дизельне, бензинове, газове, электричне».
Виробник (виробник, країна)

8. БД «Каталог програмного забезпечення підприємства»
Каталог (код_розділу, назва_розділу, код_зовнішнього_розділу, опис_розділу)
Розробники (код_розробника, назва, URL_розробника, e_mail, опис_розробника)
ПрограмніПродукти (код_ПП, код_розділу, назва_ПП, код_розробника, дата_придбання, кількість_ліцензій, початок_ліценз_періоду, кінець_ліценз_періоду, опис_ПП)
ОбліковіОдиниці (код_ОО, код_ПП, назва_ОО, тип_носія)

9. БД «Новини на сайтах»
Каталог (код_розділу, назва_розділу, код_зовнішнього_розділу, опис_розділу)

ДжерелаНовин (код, ім’я, адреса, URL, e_mail, опис)
Новини (код_новини, код_розділу, код_джерела, текст_новини, дата_розміщення)
ЧитанняНовин (код_запису, дата_час_читання, код_новини, ip_адреса_читача)

10. БД «Кадровий склад підприємства»

Співробітники (код_співробітника, ім’я, дата_народження, адреса, фото, стать)
Позиції (код_позиції, назва)
Відділи (код_відділу, назва, код_співробітника_керівника)
ПозиціїВідділи (код, код_відділу, код_позиції, особливості)
СпівробітникиПозиції (код_співробітника, код_позиції)

11. БД «Каталог фірм-виробників товарів»

Фірми (код_фірми, назва, країна, адреса, web_сайт, e_mail, телефон)
Спеціалізації (код_спец, назва, код_зовнішньої_спец)
ФірмиСпеціалізації (код_фірми, код_спец)
Товари (код_товару, одиниці_виміру, назва, характеристика, код_спец, код_фирми, ціна)

12. БД «Авіаперевезення»

Літаки (код_літака, назва, тип_двигуна, крейсерська_швидкість, макс_висота, довжина_ЗПС, дальність_польоту, виробник), де значення поля «тип_двигуна» обираються зі списку «турбореактивний, турбогвинтовий»
Аеропорти (код_аеропорту, назва, країна, місто, макс_довжина_пассажирської_ЗПС, макс_довжина_вантажної_ЗПС)
Авіаперевізники (код, назва, країна, місто, Web-сайт)
Рейси (номер_рейсу, код_авіаперевізника, код_літака, час_відльоту, час_прибуття, код_аеропорту_відльоту, код_аеропорту_прибуття, тип_рейсу), де значення поля «тип_рейсу» обираються зі списку «внутрішній, міжнародний»
СтанРейсів (номер_рейсу, код_авіаперевізника, дата, стан), де значення поля «стан» обираються зі списку «скасовано, затримано, відлетів за розкладом, прибув за розкладом».

13. БД «Водоймища світу»

ТипиВодоймищ (код_типу, назва), де значення поля «назва» обираються зі списку «річка, море, озеро, океан»
Території (код_території, назва, континент), де значення поля «континент» обираються зі списку «Євразія, Африка, Північна Америка, Південна Америка, Австралія, Антарктида», а назви територій можуть бути як назвами стран, так і назвами окремих земель (наприклад, в Антарктиді є земля Королеви Мод)

ВодоймищаСвіту (код_водоймища, назва, код_типу_водоймища, площа, протяжність, солоність, код_головного_водоймища), де значення поля «солоність» обираються зі списку «солоний, прісний»
ТериторіїВодоймища (код_водоймища, код_території, площа_акваторії, середня_глибина)

14. БД «Світові рибні ресурси»

Водоймища (код_водоймища, назва, тип_водоймища, площа, солоність), де значення поля «тип_водоймища» обираються зі списку «річка, море, озеро, океан»
Риба (код_риби, назва_сімейства, тип_місця_існування), де значення поля «тип_місця_існування» обираються зі списку «морська, річкова, прохідна»
ВодоймищаСвіту (код_місця, код_водоймища, країна, назва_акваторії)
МісцяІснування (код_риби, код_місця)

15. БД «Тварини і рослини, занесені в Червону книгу»

ТвариниРослини (код_запису, назва, клас, царство), де значення поля «клас» обираються зі списку «ссавці, птахи, плазуни, рослини, мохи і лишайники», а значення поля «царство» обираються зі списку «тварини, рослини»
ПриродніЗони (код_зони, назва, середня_температура_влітку, середня_температура_взимку середньорічна_кількість_опадів), де значення поля «назва» обираються зі списку «арктичні пустелі, тундра, лісотундра, тайга, змішані ліси, широколистяні ліси, лісостепи, степи, напівпустелі, пустелі, субтропічні ліси, мусонні ліси, високогірні ліси»
Відповідності (код_відповідності, код_зони, код_запису)
ЧервоніКниги (код_публікації, код_відповідності, статус, назва_книги, країна, автори_публікації, номер_тома, сторінки, видавництво, рік), де значення поля «статус» обираються зі списку «знакаючі, рідкісні, такі, що скорочуються, невизначені, відновлені»

16. БД «Компоненти друкованих плат»

Компонент (код_компонента, назва, опис, вартість, тип)
Інтерфейс (код_інтерфейсу, назва, тип_інтерфейсу), де значення поля «тип_інтерфейсу» обираються зі списку «цифровий, аналоговий»
СкладаєтьсяЗ (код_компонента, код_складеного_компонента, кількість)
КомпонентІнтерфейс (код_компонента, код_інтерфейсу, тип_використання), де значення поля «тип_використання» обираються зі списку «тільки вхідні дані, тільки вихідні дані, двонаправлений, допоміжний»

17. БД «Шахові партії»

Партія (код_партії, назва, гравець_білими, гравець_чорними, результат)
Позиції (позиція), де значення поля «позиція» обираються зі списку «A1,…A8,B1,…,C1,…,D1,…,E1,…,F1,…,G1, …,H1, …H8»
Фігури (код_фігури, назва, колір), де значення поля «колір» обираються зі списку «ч,б»

Хід (код_партії, номер_ходу, позиція_до, позиція_після, фігура, дія, взята_фігура), де значення поля «дія» обираються зі списку «перехід, шах, мат, рокіровка, взяття фігури»

18. БД «Гра на біржі»

Акції (код_акцій, назва, емітент, адреса_емітента, випущено_акцій, номер_випуску, ціна_за_акцію)

Брокери (код_брокера, прізвище, контора, телефон, спосіб_гри), де значення поля «спосіб_гри» обираються зі списку «на підвищення, на пониження, всі способи»

Лоти (код_лота, код_акцій, код_брокера, дата_початку_торгів, дія, кількість_акцій, стартова_ціна_за_акцію), де значення поля «дія» обираються зі списку «продаж, купівля»

Торги (код_лота, дата_пропонування_ціни, час_пропонування_ціни, код_брокера, ціна_за_акцію, результат_торгів), де значення поля «результат_торгів» обираються зі списку «торги продовжено, продано»

19. БД «Call-центр»

Клієнти (номер_телефону, тарифний_план)

Дзвінки (код_дзвінка, дата, час, номер_телефону)

ЛанцюжкиВикликів (код_дзвінка, номер_у_послідовності, код_теми)

Теми (код_теми, опис, код_зовнішньої_теми)

ДзвінкиОператору (код_дзвінка, номер_оператора, запитання, відповідь)

20. БД «Розсилка новин»

Передплатники (код, email, ім’я)

Розділ (код_розділу, опис)

ПовʼязаніРозділи (код_розділу, код_повʼязаного_розділу)

Підписка (код, код_розділу, дата_підписки, дата_відписки, отримання_повʼязаних_розділів), де значення поля «отримання_повʼязаних_розділів» обираються зі списку «так, ні»

21. БД «Послуги оператора мобільного зв'язку»

Клієнти (рахунок_клієнта, тарифний_план, діє_до, номер_телефону)

Послуги (код_послуги, назва, категорія), де значення поля «категорія» обираються зі списку «основні послуги, додаткові послуги, міжнародні вихідні дзвінки, границя попередження»

КлієнтиПослуги (рахунок_клієнта, код_послуги, дата_активації, дата_відключення)

ПерсональнаІнформація (рахунок_клієнта, ПІБ, адреса, регіон)

22. БД «Мережі мобільного зв’язку»

Оператори (код_оператора, назва, юридична_адреса)

ТочкиПокриття (код_точки, назва, регіон), де значення поля «регіон» обираються зі списку «захід, схід, північ, південь, центр»

ПрисутністьОператора (код_оператора, код_точки, кількість_стільників)
Роумінг (код_оператора, код_роумінг_оператора)
РоумінгОператори (код_роумінг_оператора, назва, країна)

23. БД «Оплата послуг мобільного зв'язку»

Клієнти (рахунок_клієнта, тарифний_план, дата_активації, номер_телефона, ПІБ, адреса, регіон)
БалансиКлієнтів (рахунок_клієнта, місяць, рік, залишок), де значення поля «місяць» обираються зі списку «1,2,3,4,5,6,7,8,9,10,11,12»
Платежі (рахунок_клієнта, місяць, рік, тип_платежу, сума), де значення поля «тип_платежу» обираються зі списку «авансові, платежі ділерам, скретч картки»
Борги (рахунок_клієнта, місяць, рік, код_послуги, сума)
Послуги (код_послуги, назва, категорія), де значення поля «категорія» обираються зі списку «вхідні, абонентська плата, вихідні всередині мережі, вихідні на інші мережі, вихідні міжнародні дзвінки, вхідні міжнародні дзвінки, послуги SMS»

24. БД «Банківські депозитні рахунки»

Клієнти (код_клієнта, назва, адреса, номер_телефона),
Рахунки (номер_рахунка, відділення_банку, дата_відкриття, дата_закриття, баланс, валюта, код_клієнта), де значення поля «валюта» обираються зі списку «UAH, EUR, USD»
Операції (номер_рахунка, дата_час_операції, сума, дія), де значення поля «дія» обираються зі списку «поповнення, закриття, зняття готівки»
ХарактеристикиРахунків (номер_рахунка, строк_дії, періодичність_нарахування_процентів, процент_річних)

25. БД «Кредитні карти»

Власники (код_власника, назва, адреса, номер_телефона),
КредитніКартки (номер_карти, тип_картки, валюта_рахунка, дата_відкриття, дата_закриття, перший_внесок, баланс, код_власника), де значення поля «валюта» обираються зі списку «UAH, EUR, USD», а значення поля «тип_карти» обираються зі списку «EC/MC, Visa»
Операції (номер_картки, дата_операції, час_операції, сума, валюта, дія), де значення поля «дія» обираються зі списку «оплата товарів, зняття готівки»
ХарактеристикиКарток (номер_картки, незнижуваний_залишок, кредитний_ліміт, розмір_овердрафту)

Контрольні питання:

1. Необхідність використання первинних ключів.
2. Необхідність використання індексів.
3. Як створити зв'язок 1:M між таблицями?
4. Як створити зв'язок 1:1 між таблицями?

5. Знайти еквівалент в мові ANSI SQL для обов’язкового поля; для індексованого поля; для значення за умовчанням; для підпису поля; для умови на значення.

Додаток Б

Типи таблиць MySQL

Тип MyISAM зазвичай використовується для таблиць-довідників. Його особливості є такими:

- транзакції не підтримуються;
- максимальний розмір диску: 256 Тб;
- повнотекстовий пошук;
- блокування рівня таблиці;
- не працює у кластері;
- не підтримуються зовнішні ключі, обмеження цілісності;
- підтримується реплікація;
- максимальна кількість індексів: 64;
- максимальна кількість записів: 2^32;
- максимальна довжина ключа: 1000 байт;
- максимальна сумарна довжина полів VARCHAR та CHAR: 64 Кб.

Тип таблиць MyISAM підтримує три формати фізичного збереження: статичний (fixed-length), динамічний (dynamic), стиснутий (compressed).

Статичний формат використовується автоматично, якщо в таблиці немає стовпців типів VARCHAR, VARBINARY, BLOB, TEXT.

Динамічний формат використовується, якщо хоча б один стовпець таблиці має тип VARCHAR, VARBINARY, BLOB, TEXT. Пусті рядки та нулі (0) не займають місця на диску.

Стиснутий формат рекомендується для повільних носіїв даних, він створюється за допомогою утиліти myisqmpack та не передбачає оновлення даних (тобто, створюється для даних, які будуть лише зчитуватися).

Тип InnoDB використовується для великих таблиць, для яких також важливим є надійність оновлення даних в транзакціях (високонавантажені додатки, фінансові транзакції).

- максимальний розмір диску: 64 Тб;
- повна підтримка транзакцій (4 рівня ізоляції);
- блокування рівня запису;
- повнотекстовий пошук не підтримується;
- безпечний для транзакцій;
- індекси кластеризуються для типових запитів;
- підтримка зовнішніх ключів;
- підтримка Raw Disk, тобто організація доступу до даних без використання файлової системи;
- може використовуватися на операційних системах з обмеженням на розмір файлу.

Тип таблиць MERGE застосовується для організації ідентичних таблиць в одну віртуальну таблицю. Його властивості:

- об’єднання однакових таблиць в одну;
- таблиці повинні мати однакову структуру, порядок стовпців має бути однаковим;
- таблиці можуть бути у інших база даних;
- не підтримується повнотекстовий пошук;
- DROP TABLE не видаляє таблиць, які використовувалися у таблиці типу MERGE;
- чутливий до змін структури таблиць.

Тип таблиць HEAP (MEMORY) використовується для невеличких за обсягом тимчасових таблиць. Його особливості:

- не підтримуються транзакції;
- блокування рівня таблиці;
- підтримується реплікація;
- максимальна довжина ключа: 500 байт;
- дані знищуються (а сама таблиця залишається) при зупинці сервера;
- формат збереження – завжди fixed-length.

Тип таблиць ARCHIVE використовується для ведення різноманітних журналів, у тому числі журналів сервера. Його особливості:

- немає обмеження на максимальний розмір диску;
- блокування рівня запису;
- не застосовуються операції DELETE, UPDATE, ORDER BY, REPLACE;
- не використовується тип даних BLOB;
- SELECT виконується дуже довго.