УДК 004:378 DOI: 10.30929/1995-0519.2019.5.110-115

ВИБІР МЕТОДУ ДЛЯ ПРИСКОРЕННЯ ПОШУКУ ІНФОРМАЦІЇ В РЕЛЯЦІЙНІ БАЗІ ДАНИХ

О. О. Москаленко
Кременчуцький національний університет імені Михайла Остроградського
вул. Першотравнева, 20, м. Кременчук, 39600, Україна.

Важливим фактором при роботі з базами даних є можливість організації належного пошуку. Бази даних мають різні моделі даних та системи управління базами даних, структуру організації баз даних, все це впливає на організацію, якість та швидкість пошуку. В даній роботі було розглянуто основні моделі даних і більш детально реляційну модель даних та реляційну алгебру. Було досліджено різницю між повнотекстовим пошуком та пошуком за допомогою операторів порівняння в реляційній моделі даних за допомогою реляційної алгебри. Також виконано порівняння структури організації даних і яким чином це впливає на пошук. Виявлено, що вибір методу пошуку даних впливає на швидкість пошуку, тобто повнотекстовий пошук значно прискорює пошук, особливо в великих масивах даних. Структура організації даних в реляційній моделі також впливає на швидкість пошуку, в залежності від відносин і зв'язків між даними. Це грає суттєву роль при роботі з великими масивами даних.

Ключові слова: Бази даних, модель даних, СУБД, реляційна алгебра, повнотекстовий пошук, структура баз даних, MySQL.

ВЫБОР МЕТОДА ДЛЯ УСКОРЕНИЯ ПОИСКА ИНФОРМАЦИИ В РЕЛЯЦИОННОЙ БАЗЕ ДАННЫХ

А. А. Москаленко
Кременчугский национальный университет имени Михаила Остроградского
ул. Первомайская, 20, г. Кременчуг, 39600, Украина.

Важным фактором при работе с базами данных является возможность организации надлежащего поиска. Базы данных имеют различные модели данных и системы управления базами данных, структуру организации баз данных, все это влияет на организацию, качество и скорость поиска. В данной работе были рассмотрены основные модели данных и более подробно реляционную модель данных, и реляционную алгебру. Было исследовано разницу между полнотекстовым поиском и поиском с помощью операторов сравнения в реляционной модели данных с помощью реляционной алгебры. Также выполнено сравнение структуры организации данных и, каким образом это влияет на поиск. Выявлено, что выбор метода поиска данных влияет на скорость поиска, то есть полнотекстовый поиск, значительно ускоряет поиск, особенно в больших массивах данных. Структура организации данных в реляционной модели также влияет на скорость поиска, в зависимости от отношений и связей между данными. Это играет существенную роль при работе с большими массивами данных.

Ключевые слова: Базы данных, модель данных, СУБД, реляционная алгебра, полнотекстовый поиск, структура баз данных, MySQL.

АКТУАЛЬНІСТЬ РОБОТИ. Основою будь-якої бази даних являється модель даних. Якщо трактувати базу даних як набір заданих фактичних даних, то з цих фактичних даних можна логічно вивести інші фактичні дані [1]. Для представлення логічної структури набору таких даних та взаємозв'язків між ними використовують поняття модель даних. Тобто модель даних – це абстрактне, логічне визначення об'єктів, операторів та інших елементів бази даних, які в цілому складають абстрактну машину доступу до даних, з якою взаємодіє користувач [1].

Структура бази даних відображає в структурованому упорядкованому вигляді логічний набір даних та взаємозв'язки між ними. Взаємозв'язки між логічними об'єктами бувають трьох видів: «один до одного», «один до багатьох» та «багато до багатьох». В залежності від взаємозв'язків між об'єктів ми маємо такі моделі даних: реляційну, ієрархічну, мережеву, об'єктно-зорієнтовану та об'єктно-реляційну. Широко використовуються з них об'єктно-зорієнтовані, об'єктно-реляційні і реляційні моделі даних.

Об'єктна (об'єктно-зорієнтована) модель бази даних – модель даних в якій дані модулюються в вигляді об'єктів, їх атрибутів та класів. Така модель ґрунтується на об'єктно-зорієнтованих мовах програмування. Об'єктно-зорієнтовані бази даних зазвичай використовуються, коли потрібна високопродуктивна обробка даних, що мають складну структуру.

Парадигма об'єктно-реляційної моделі даних об'єднує основні переваги реляційної моделі даних та деякі, успадковані від об'єктно-зорієнтованої моделі даних. Варто сказати, що «об'єктність» в об'єктно-реляційної моделі відрізняється від об'єктно-зорієнтованої моделі – об'єктом в об'єктно-реляційній моделі являються дані, а не семантика зв'язку реального світу. Це дозволяє, з одного боку, використовувати механізми спадкування і перевизначення, звернення до об'єктів із застосуванням спеціалізованих методів, а з іншого — вирішувати складні аналітичні задачі, пов'язані з логічним аналізом значень атрибутів.

Сама розповсюджена модель даних – реляційна. В ній є можливість реорганізовувати інформаційну базу тому що в ній відсутні відмінності між об'єктами та зв'язками між ними.

У реляційній моделі бази даних, зв'язки між елементами даних представляються у вигляді двомірних таблиць, які називаються відносинами. Таблиця може мати декілька стовбців, в яких встановлюється тип даних. Дані двох таблиць пов'язуються загальними стовпцями. Пошук даних в ній відбувається за значенням ключових атрибутів.

Реляційна модель даних заснована на математичному понятті «відношення» з теорії множин, фізичним відображенням якого є таблиця. В будь-якій реляційній системі управління базами даних (СУБД) припускається, що користувач сприймає базу даних як набір таблиць. В реляційній моделі «відношення» використовуються для зберігання інформації про об'єкти представленні в базі даних. Зазвичай «відношення» представлено у вигляді двовимірної таблиці де рядки мають окремі записи, а стовбці – атрибути [2].

В реляційній моделі бази даних, використовуючи відносини зв'язку та мову реляційної алгебри можна здійснювати вибір будь-якої підмножини інформації, за рядками, стовпцями, або іншими ознаками.

Реляційна алгебра базується на теорії множин та є основною логікою роботи реляційної моделі даних. Це мова операцій, які на основі відношень дозволяють створювати інше відношення без зміни вихідних відношень [2]. І вихідні відношення і результат є відношеннями, тому результат може стати вихідним відношенням для других операцій.

Основними операціями над відношеннями є: об'єднання, перетин, різниця, декартовий добуток, вибірки (або скорочення), проекція, з'єднання та поділ.

Об'єднання двох множин в математиці є множина всіх елементів, що належать, або до однієї з них, або до обох заданих множин, але результат не є відношенням, тому що відношення не можуть мати кортежі різних типів. Тому об'єднання в реляційній алгебрі не повність відповідає об'єднання в математиці. Якщо дані відношення одного типу *a* і *b*, то об'єднанням цих відношень *a UNION b* є відношення того ж типу, які складається з кортежів *t*, де *t* рівняється деякому кортежу із *a* або *b* або з обох відношень [1].

Якщо дані відношення одного типу *a* і *b*, то перетином цих відношень *a INTERSECT b* є відношення того ж типу, які складається з кортежів *t*, де *t* присутнє одночасно і в *a* і в *b* [1].

Якщо дані відношення одного типу *a* і *b*, то різницею цих відношень *a MINUS b* є відношення того ж типу, які складається з кортежів *t*, де *t* присутнє тільки в *a*, але не в *b* [1].

Декартовим добутком в реляційній алгебрі кожна впорядкована пара замінюється одним кортежем, що є об'єднанням двох розглянутих кортежів.

Операція вибірки в реляційній алгебрі працює з одним відношенням *t* та визначає відношення результату, яке містить тільки ті кортежі відношення *t*, які задовольняють заданій умові. Наприклад, якщо є множина A_1 , де A_1*={2,4}* і задана умова *t<3*, то операція вибірки буде мати вигляд A_1 *WHERE t<3*, а результатом буде *t={2}*.

Операція проекції працює з одним відношенням *t* та визначає відношення результату, що містить вертикальну підмножину відношення *t*, створюване за допомогою вилучення значень зазначених атрибутів та вилучення із результату рядків дублікатів.

Отже основним призначенням реляційної алгебри є забезпечення можливості складання реляційних виразів.

Метою даної роботи було дослідження і виявлення залежності пошуку інформації у базі даних від вибору методу пошуку, використовуючи реляційну алгебру для організації запитів і від типу організації структури бази даних.

МАТЕРІАЛ І РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ. Між фізичною базою даних, тобто даними які зберігаються на сервері, та користувачами системи знаходиться рівень програмного забезпечення СУБД [3]. Це програмне забезпечення, завдяки якому користувачі мають можливість створювати та підтримувати базу даних, та розподіляти рівень доступу до бази даних. Всі запити до бази даних, засоби додавання, вибірки та оновлення даних обробляються СУБД. Кожна СУБД управляє однією з моделей даних.

Розглянемо більш детально реляційну СУБД на прикладі порівняння пошуку. Більшість реляційних СУБД підтримують методи повнотекстового пошуку по базі даних [4, 5]. Якщо розглядати пошук по базі даних на прикладі реляційної системи управління базами даних MySQL то існує два варіанти організації пошуку: за допомогою операторів порівняння та за допомогою повнотекстового пошуку. На відміну від операторів порівняння, повнотекстовий пошук базується на створенні відповідного повнотекстового індексу текстових даних, який являє собою деякий словник слів, які зустрічаються у цих даних. Для порівняння цих двох методів пошуку, створимо дві таблиці в базі даних, та організуємо пошук кожним методом. Створимо таблицю «lessons_fulltext» з повнотекстовим індексом, яка має поля «id» для нумерації даних, поле «name» для назви лекційного матеріалу, поле «directions_ID» для номеру категорії лекції, поле «info» для матеріалу лекції, повнотекстовий індекс для індексування поля name та info. Також створимо таблицю «lessons» для пошуку за допомогою операторів порівняння, що має такі ж поля, що й таблиця «lessons_fulltext». Приклад створення таблиці з повнотекстовим індексом:

```
CREATE TABLE IF NOT EXISTS ` lessons_fulltext ` (
`id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(150) NOT NULL,
`directions_ID` mediumint(8) unsigned DEFAULT `0`,
`info` text,
PRIMARY KEY (`id`),
KEY `les_indx1` (`directions_ID`),
FULLTEXT KEY `index` (`name`, `info`),
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

Приклад створення таблиці для пошуку з оператором порівняння:

```
CREATE TABLE IF NOT EXISTS ` lessons` (
`id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(150) NOT NULL,
`directions_ID` mediumint(8) unsigned DEFAULT `0`,
`info` text,
```

PRIMARY KEY (`id`),
) CHARSET=utf8;

Дані приклади взяті зі структури бази даних дистанційної системи на базі e-Front, в якій таблиця *lessons* зберігає в собі лекції предмету [6, 7].

З точки зору реляційної алгебри ми маємо дві множини *lessons_fulltext {id, name, directions_ID, info, index }* та *lessons {id, name, directions_ID, info }*.

Щоб порівняти пошук, заповнимо ці дві таблиці даними та виконаємо пошук по кожній таблиці та порівняємо час виконання. В кожній таблиці будемо шукати записи в яких є слово «database». Тобто з точки зору реляційної алгебри виконаємо операцію вибірки *lessons { info } WHERE info = database* та *lessons_fulltext { index } WHERE index = database*.

Виконаємо наступний запит до бази даних:

set profiling=1;
*SELECT * FROM `lessons_fulltext` WHERE MATCH (name, info) AGAINST ('*database*' IN BOOLEAN MODE); // Запит з використанням повнотекстового пошуку*
*SELECT * FROM `lessons` WHERE lessons.info LIKE '%database%'; // запит з використанням оператора порівняння LIKE*
show profiles;

Якщо розглядати запити з точки зору реляційної алгебри, в першому випадку з повнотекстовим пошуком ми здійснюємо пошук по індексу множини lessons_fulltext{index}='databese', що містить в собі дані елементів «name» та «info», а в другому випадку ми здійснюємо пошук по елементу множини lessons {info} = 'databese'. Тобто запит з оператором порівняння порівнює кожен рядок на відповідність до умови запиту, а повнотекстовий запит формує повнотекстовий індекс, який зберігає в собі всі пошукові дані документа та в яких місцях документу вони зустрічаються. Такий підхід значно збільшує швидкість пошуку, ніж сканування всього вмісту таблиць бази даних. Результат виконання даного запиту показаний на рис. 1.

```
| Query_ID | Duration   | Query
     |
+----------+------------+-------------------------------------------------------------------------------------------------
-----+
|        1 | 0.00010475 | SELECT * FROM `lessons_fulltext` WHERE MATCH (name, info) AGAINST ('*database*' IN BOOLEAN
MODE) |
|        2 | 0.00011525 | SELECT * FROM `lessons` WHERE lessons.info LIKE '%database%'
     |
+----------+------------+-------------------------------------------------------------------------------------------------
-----+
```

Рисунок 1 – Результат пошуку двох запитів

Як ми бачимо з результату виконання запиту, повнотекстовий пошук працює швидше ніж пошук з оператором порівняння. Чим більше даних в таблиці, тим значніше відрізняється швидкість пошуку по базі даних. Повнотекстовий пошук в СУБД MySQL окрім своєї швидкої роботи, на відміну від пошуку за допомогою операторів порівняння, має можливість ранжувати знайдені данні, тобто можна знаходити самі релевантні записи для пошукового запиту. Також за допомогою повнотекстового пошуку, можна здійснювати пошук з урахуванням морфології.

Пошук по базі даних завдяки операторам порівняння та повнотекстового пошуку мають свої переваги і недоліки, які наведені в табл. 1 [8–10].

Таблиця 1 – Переваги та недоліки повнотекстового пошуку та пошуку завдяки операторам порівняння

	Оператор порівняння	Повнотекстовий пошук
Переваги	Можливість сортування результатів пошуку; Пошук можна здійснювати майже по всім типам полів.	Підтримка морфології; Видача результатів пошуку за релевантністю; Наявність модифікаторів; Стоп-слова для пошуку; Можливість налаштування пошуку.
Недоліки	Відсутність підтримки морфології ; Відсутність модифікаторів; Пошук здійснюється перебором всіх рядків таблиці.	Відсутність можливості сортування; Підтримка пошуку тільки для полів varchar та text; Ресурсномісткий процес; Для повнотекстового індексу додавання даних в таблицю відбувається значно довше.

При виборі пошуку по базі даних, доцільно враховувати переваги та недоліки кожного методу.

Окрім методу пошуку, також має значення структура організація бази даних. Наприклад, нам потрібно створити дві таблиці для курсів та лекцій, таблиці повинні мати зв'язок «один до багатьох» – один курс містить в собі багато лекцій. Реалізувати структури цієї бази даних ми можемо наступними шляхами:

- створити дві таблиці з використанням «foreign key» для реалізації зв'язку «один до багатьох»;
- створити дві таблиці без використання «foreign key» та створити третю допоміжну таблицю з id таблиці курсів та id таблиці лекції для їх зв'язку.

Розглянемо кожен метод. Для створення таблиць з використанням «foreign key» можемо виконати наступний SQL запит:

```
CREATE TABLE IF NOT EXISTS `courses1` (
`id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(150) NOT NULL,
PRIMARY KEY (`id`)
) CHARSET=utf8;

CREATE TABLE IF NOT EXISTS `lessons1` (
`id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
`id_courses` mediumint(8) unsigned NOT NULL,
`name` varchar(150) NOT NULL,
`info` text,
PRIMARY KEY (`id`),
FOREIGN KEY(`id_courses`) REFERENCES courses1(`id`)) CHARSET=utf8;
```

Після виконання даного SQL запиту створяться дві таблиці об'єднаних між собою зв'язком «один до багатьох». Структура даної бази даних наведена на рис. 2.

Рисунок 2 – Зв'язок «один до багатьох» з використанням «foreign key»

Для використання другого методу створення структури бази даних виконаємо наступний SQL запит:

```
CREATE TABLE IF NOT EXISTS `courses2` (
`id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(150) NOT NULL,
PRIMARY KEY (`id`)
) CHARSET=utf8;

CREATE TABLE IF NOT EXISTS `lessons2` (
`id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
`name` varchar(150) NOT NULL,
`info` text,
PRIMARY KEY (`id`)
) CHARSET=utf8;

CREATE TABLE IF NOT EXISTS `courses_to_lessons` (
`id` mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
`id_courses` mediumint(8) NOT NULL,
`id_lessons` mediumint(8) NOT NULL,
PRIMARY KEY (`id`)
) CHARSET=utf8;
```

Після виконання даного SQL запиту створяться три таблиці об'єднаних між собою третьою допоміжною таблицею. Структура даної бази даних наведена на рис. 3.

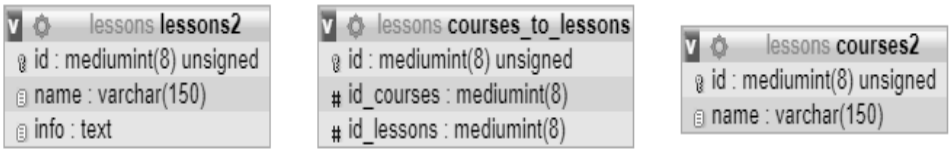
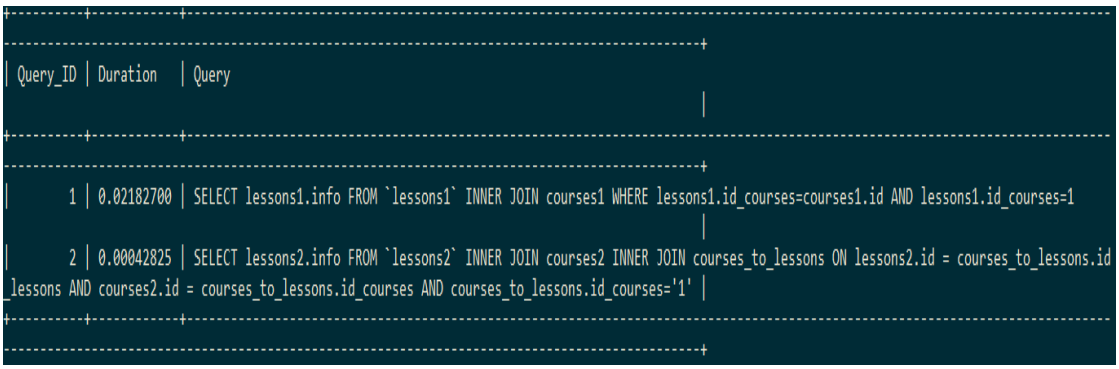


Рисунок 3 – Зв'язок без з використанням «foreign key» за допомогою допоміжної таблиці

В дистанційній системі навчання e-Front таблиця *courses* зберігає в собі предмети, а таблиця *lessons* зберігає в собі лекції предмету.

Останній метод організації структури таблиць використовує зв'язок «багато до багатьох» – за допомогою допоміжної таблиці. В нашому випадку два різних курси не можуть містити в собі одні й ті ж самі лекції, але така організація структури нам підходить тому, що завдяки програмному коду, що буде звертатися до бази даних можна уникнути дублювання даних. Таким чином, ми маємо два варіанти організації структури бази даних лекцій та курсів.

Для порівняння цих двох видів організації бази даних, заповнимо таблиці курсів та лекції тестовими даними та виконаємо пошук по базі даних для кожної структури. Виконаємо два аналогічних SQL запиту для кожної структури бази даних, та порівняємо швидкість виконання запиту (рис. 4).



Query_ID	Duration	Query
1	0.02182700	SELECT lessons1.info FROM `lessons1` INNER JOIN courses1 WHERE lessons1.id_courses=courses1.id AND lessons1.id_courses=1
2	0.00042825	SELECT lessons2.info FROM `lessons2` INNER JOIN courses2 INNER JOIN courses_to_lessons ON lessons2.id = courses_to_lessons.id_lessons AND courses2.id = courses_to_lessons.id_courses AND courses_to_lessons.id_courses='1'

Рисунок 4 – Швидкість виконання двох аналогічних запитів до таблиць з різною організацією структури БД

Якщо розглянути ці два варіанта з точки зору пошуку по базі даних, то в першому випадку, запит буде перебирати всі рядки бази даних, а в другому випадку, завдяки допоміжної таблиці, пошук буде відразу по необхідним рядкам. Чим більше рядків та даних буде в базі даних, тим більше буде відрізнятися швидкість виконання запитів. Для малої бази даних швидкість майже не буде відрізнятися, тому останній метод доцільніше використовувати для бази даних з великою кількістю даних. Окрім правильної організації структури бази даних, на швидкість також впливає побудова SQL запита.

ВИСНОВКИ. Отже в роботі було розглянуто основні поняття та види моделей даних і реляційну модель з точки зору реляційної алгебри. Досліджено різницю між повнотекстовим пошуком та пошуком за допомогою операторів порівняння в реляційній моделі даних за допомогою реляційної алгебри. Також виконано порівняння структури організації даних. Виявлено, що вибір методу та організація структури бази даних впливає на швидкість пошуку, особливо це суттєво для великих масивів даних.

ЛІТЕРАТУРА

1. Date K. J. An Introduction to database systems. 8th edition. New York: Addison Wesley, 2005. 1328 p.

2. Connolly T. M., Begg C. E. Database systems: A practical approach to design, implementation and management (4th Edition), New York: Addison Wesley, 2004. 1440 p.

3. Hryhorova T., Moskalenko O. O. Use of Information Technologies to Improve Access to Information in E-Learning Systems. Recent Developments in Data Science and Intelligent Analysis of Information. Proceedings of the XVIII International Conference on Data Science and Intelligent Analysis of Information, June 4–7, 2018, Kyiv, Ukraine, P. 206–215.

4. Hryhorova T., Moskalenko O. O. Enhanced Access to Training Information in E-learning Systems. Proceedings of 41 International convention on information and communication technology, electronics and microelectronics "MIPRO 2018", Opatija (Croatia), 21-25 травня, 2018. P. 1070–1074.

5. Москаленко О. О., Григорова Т. А. Шляхи розширення структури даних в системах електронного навчання. Матеріали міжнародної наукової конференції «Сучасні проблеми математичного моделювання, обчислювальних методів та інформаційних технологій», Рівне, 2-4 березня 2018. С. 268–270.

6. Павлов Д. Кластерные СУБД. Jet Info №12. 2009. Режим доступу до ресурсу: http://www.jetinfo.ru/stati/klasternye

7. Хендерсон К. Профессиональное руководство по SQL Server: структура и реализация. Перев. с англ. М.: ООО «Вильямс», 2006. 1044 с.

8. Поиск: fulltext или like? [Електр. Ресурс]. – Режим доступу до ресурсу: https://habr.com/ru/post/25646/.

9. Lemahieu W., Broucke vanden Seppe, Baesens B. Principles of Database Management: The Practical Guide to Storing, Managing and Analyzing Big and Small Data, Cambrige University Press. 2018. 808 p.

10. Kroenke D. M., Auer D. J. Database Processing: Fundamentals, Design, and Implementation, 14th ed. Pearson Education Ltd., 2016. 638 p.

THE CHOICE OF METHOD TO SPEED UP THE SEARCH OF INFORMATION IN A RELATIONAL DATABASE

O. Moskalenko

Kremenchuk Mykhailo Ostrohradskyi National University

vul. Pershotravneva, 20, Kremenchuk, 39600, Ukraine

Purpose. The article aims to research and identify the dependence of searching information in the database on the choice of search method, using relational algebra to organize queries and the type of organization of the database structure. **Methodology**. Research and identification of the dependence of search speed on the search method, organization of queries and the type of organization of the database structure is important for organizing the structure and methods of data search. An important factor when working with databases is the ability to organize a proper search. There were observed different databases that have different data models and database management systems, database structure, all of which affects the organization, quality and speed of search. More detailed were observed the relational data model and relational algebra, which is the main logic of the relational data model. Relationship operations are used

to build query to a relational database. **Results**. The difference between full-text search and search using comparison operators in a relational data model using relational algebra was investigated. To do this, in the same table was performed a search by using two different methods. Also there was made a comparison of the structure of the data organization and how it affects the search. There were created tables that differ by using the "outside key" in some of them, while others used an additional table contained the ID, formed from the two previous tables. In the second case, the search was performed by the necessary lines straight away, which reduces the search time. The more records in the database tables, the more the query execution time exceeds. **Originality**. It was revealed that the choice of the data search method affects the search speed, that is, full-text search, significantly speeds up the search, especially in large data arrays. The structure of data organization in the relational model also affects the speed of the search, depending on the relationships between the data. **Practical value.** When working with large data arrays, the use of full-text search in a relational database effects an important role, and special attention should be took in consideration while organizing of the data structure. However, attention should be delivered to data duplication, which is not acceptable whsle forming a database.

Key words: Databases, data model, DBMS, relational algebra, full-text search, database structure, MySQL.

REFERENCES

1. Date, K. J. (2005), An Introduction to database systems. 8th edition, Addison Wesley, New York.

2. Connolly, T. M., Begg, C. E. (2004*)*, Database systems: A practical approach to design, implementation and management (4th Edition), Addison Wesley, New York.

3. Hryhorova, T., Moskalenko, O. (2018), "Use of Information Technologies to Improve Access to Information in E-Learning Systems", Recent Developments in Data Science and Intelligent Analysis of Information. Proceedings of the XVIII International Conference on Data Science and Intelligent Analysis of Information, Kyiv, June 4–7, 2018, pp. 206–215.

4. Hryhorova, T., Moskalenko, O. (2018), "Enhanced Access to Training Information in E-learning Systems", *Proceedings of 41 International conventions on information and communication technology, electronics and microelectronics "MIPRO 2018",* Opatija (Croatia), May 21-25, 2018, pp. 1070-1074.

5. Hryhorova, T., Moskalenko, O. (2018), "Ways to expand the data structure in e-learning systems", *Proceedings of the international scientific conference "Modern problems of mathematical modeling, computational methods and information technologies"*, Rivne, March 2-4, 2018, pp. 268–270.

6. Pavlov, D. (2009), "A clustered DBMS", *Jet Info no. 12*, Mode of access to the resource: http://www.jetinfo.ru/stati/klasternye.

7. Henderson, K. (2006), Professional guide to SQL Server: structure and implementation, Pearson Education Ltd.

8. Poisk: fulltext ili like? [Search fulltext or like?] Mode of access to the resource: https://habr.com/ru/post/25646/.

9. Lemahieu, W., Broucke vanden Seppe, Baesens, B. (2018), Principles of database management: The practical guide to storing, managing and analyzing big and small data, Cambrige University Press.

10. Kroenke, D. M., Auer, D. J. (2016), Database processing: fundamentals, design, and implementation, 14th ed., Pearson Education Ltd.

Стаття надійшла 21.10.2019.