



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ

«Информатика и системы управления»

КАФЕДРА

«Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

«Классификация методов построения
индексов в базах данных»

Студент:

ИУ7-73Б

(группа)

(подпись, дата)

М. Д. Маслова

(И. О. Фамилия)

Преподаватель:

(подпись, дата)

А. А. Оленев

(И. О. Фамилия)

2022 г.

РЕФЕРАТ

Расчетно-пояснительная записка 19 с., 10 рис., 0 табл., 18 источн., 0 прил.
ИНДЕКСЫ, В-ДЕРЕВЬЯ, ХЕШ-ИНДЕКСЫ, БИТОВЫЕ ИНДЕКСЫ,
ОБУЧЕННЫЕ ИНДЕКСЫ, БАЗЫ ДАННЫХ, СИСТЕМЫ УПРАВЛЕНИЯ
БАЗАМИ ДАННЫХ

Объектом исследования является построение индексов в базах данных.

Цель работы — классификация методов построения индексов в базах данных.

В разделе 1 рассмотрено понятие индекса в базах данных и его основные свойства, а также описаны типы индексов.

В разделе 2 проведен обзор методов построения индексов на основе В-деревьев, хеш-таблиц и битовых карт, а также соответствующих обученных индексов.

В разделе 3 приведены критерии оценки качества описанных методов и проведено сравнение по этим критериям.

что-то про результат

СОДЕРЖАНИЕ

РЕФЕРАТ	3
ВВЕДЕНИЕ	5
1 Анализ предметной области	6
1.1 Основные определения	6
1.2 Типы индексов	7
2 Описание существующих методов построения индексов	9
2.1 Индексы на основе деревьев поиска	9
2.1.1 В-деревья	9
2.1.2 B^+ -деревья	11
2.1.3 Обученные индексы	11
2.2 Индексы на основе хеш-таблиц	13
2.2.1 Хеш-индексы	13
2.2.2 Обученные хеш-индексы	13
2.3 Индексы на основе битовых карт	14
2.3.1 Фильтр Блума	14
2.3.2 Обученные индексы	14
3 Классификация существующих методов	16
ЗАКЛЮЧЕНИЕ	17
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	19

ВВЕДЕНИЕ

На протяжении последнего десятилетия происходит автоматизация все большего числа сфер человеческой деятельности [1]. Это приводит к тому, что с каждым годом производится все больше данных. Так, по исследованию компании IDC (International Data Corporation), занимающейся изучением мирового рынка информационных технологий и тенденций развития технологий, объем данных к 2025 году составит около 175 зеттабайт, в то время как на год исследования их объем составлял 33 зеттабайта [2].

Для хранения накопленных данных используются базы данных (БД), доступ к ним обеспечивается системами управления базами данных (СУБД), обрабатывающими запросы на поиск, вставку, удаление или обновление. При больших объемах информации необходимы методы для уменьшения времени обработки запросов, одним из которых является построение индексов [3].

Базовые методы построения индексов используют такие структуры, как деревья поиска, хеш-таблицы и битовые карты [4]. На основе данных методов проводятся исследования по разработке новых для уменьшения времени поиска и затрат на перестроение индекса при изменении данных, а также сокращения дополнительно используемой памяти. Одно из таких исследований [5] было проведено в 2018 году, авторы которого опираясь на идею, что обычные индексы не учитывают распределение данных, предложили новый вид индексов, основанный на машинном обучении, и назвали их обученные индексы (learned indexes). За последние пять лет было проведено множество исследований [6–9] по совершенствованию обученных индексов в плане поддержки операций и улучшению производительности, поэтому в данной работе в сравнение к базовым приводятся методы построения обученных индексов.

Целью данной работы является **классификация методов построения индексов в базах данных**.

Для достижения поставленной цели требуется решить следующие задачи:

- провести анализ предметной области: дать основные определения, описать свойства индексов и их типы;
- описать методы построения индексов в базах данных;
- предложить и обосновать критерии оценки качества описанных методов и сравнить методы по предложенным критериям оценки.

1 Анализ предметной области

1.1 Основные определения

Индекс — это некоторая структура, обеспечивающая быстрый поиск записей в базе данных [10]. Индекс определяет соответствие значения атрибута или набора атрибутов — *ключа поиска* — конкретной записи с местоположением этой записи [11]. Это соответствие организуется с помощью индексных записей. Каждая из них соответствует записи в *индексируемой таблице* — таблице, по которой строится индекс — и содержит два поля: идентификатор записи или указатель на нее, а также значение индексированного поля в этой записи [12].

Индексы могут использоваться для поиска по конкретному значению или диапазону значений, а также для проверки существования элемента в таблице, однако обеспечение уменьшения времени доступа к записям в общем случае достигается за счет [11]:

- упорядочивания индексных записей по ключу поиска, что уменьшает количество записей, которые необходимо просмотреть;
- а также меньшего размера индекса по сравнению с индексируемой таблицей, сокращающего время чтения одного элемента.

В то же время индекс является структурой, которая строится в дополнение к существующим данным, то есть он занимает дополнительный объем памяти и должен соответствовать текущим данным. Последнее значит, что индекс необходимо изменять при вставке или удалении элементов, на что затрачивается время, поэтому индекс, ускоряя работу СУБД при доступе к данным, замедляет операции изменения таблицы, что необходимо учитывать [13].

Таким образом, индекс может описываться: [11]:

- *типом доступа* — поиск записей по атрибуту с конкретным значением, или со значением из указанного диапазона;
- *временем доступа* — время поиска записи или записей;
- *временем вставки*, включающее время поиска правильного места вставки, а также время для обновления индекса;
- *временем удаления*, аналогично вставке, включающее время на поиск удаляемого элемента и время для обновления индекса;
- *дополнительной памятью*, занимаемая индексной структурой.

1.2 Типы индексов

Индексы могут быть:

- кластеризованные и некластеризованные;
- плотные и разреженные;
- одноуровневые и многоуровневые;
- а также иметь в своей основе различные структуры, что описывается в следующем разделе, так как исследуется в данной работе.

В *кластеризованных* индексах логический порядок ключей определяет физическое расположение записей, а так как строки в таблице могут быть упорядочены только в одном порядке, то кластеризованный индекс может быть только один на таблицу. Логический порядок *некластеризованных* индексов не влияет на физический, и индекс содержит указатели на записи таблицы [13].

Плотные индексы (рисунок 1.1) содержат ключ поиска и указатель на первую запись с заданным ключом поиска. При этом в кластеризованных индексах другие записи с заданным ключом будут лежать сразу после первой записи, так как записи в таких файлах отсортированы по тому же ключу. Плотные некластеризованные индексы должны содержать список указателей на каждую запись с заданным ключом поиска [11].



Рисунок 1.1 – Плотный индекс

В *разреженных* индексах (рисунок 1.2) записи содержат только некоторые значения ключа поиска, а для доступа к элементу отношения ищется запись индекса с наибольшим меньшим или равным значением ключа поиска, происходит переход по указателю на первую запись по найденному ключу и далее по указателям в файле происходит поиск заданной записи. Таким образом, разреженные индексы могут быть построены только на отсортированных последовательностях записей, иначе хранения только некоторых ключей поиска

будет недостаточно, так как будет неизвестно, после записи, с каким ключом будет лежать необходимый элемент отношения [11].



Рисунок 1.2 – Разреженный индекс

Поиск с помощью неразреженных индексов быстрее, так как указатель в записи индекса сразу приводит к необходимым записям. Однако разреженные индексы требуют меньше дополнительной памяти и сокращают время поддержания структуры индекса в актуальном состоянии при вставке или удалении [11].

Одноуровневые индексы ссылаются на данные таблице, индексы же *верхнего уровня многоуровневой* структуры ссылают на индексы нижестоящего уровня [11] (рисунок 1.3).

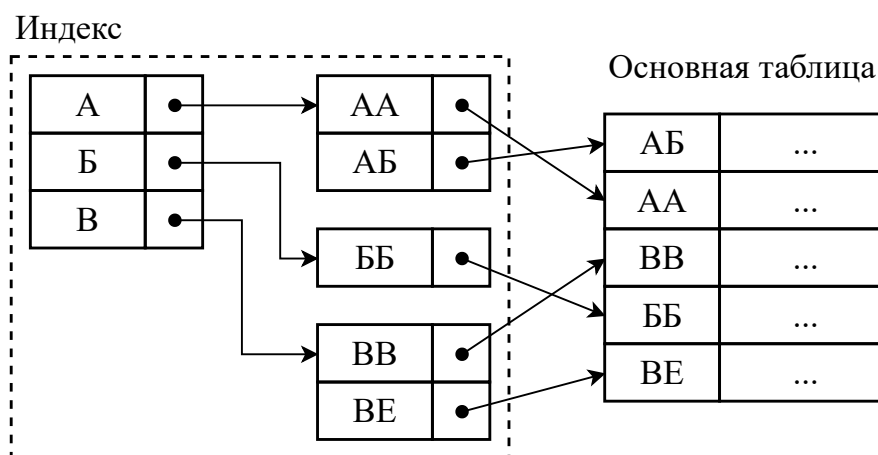


Рисунок 1.3 – Многоуровневый индекс

2 Описание существующих методов построения индексов

Как было сказано выше индексы обеспечивают быстрый поиск записей, поэтому в их основе лежат структуры, предназначенные для решения этой задачи. По данным структурам индексы подразделяются на

- индексы на основе деревьев поиска,
- индексы на основе хеш-таблиц,
- индексы на основе битовых карт.

2.1 Индексы на основе деревьев поиска

Дерево поиска — иерархическая структура, используемая для поиска записей, которая осуществляет работу с отсортированными значениями ключей и в которой каждый переход на более низкий уровень иерархии уменьшает интервал поиска. При использовании деревьев поиска для построения индексов необходимо учитывать, что требуется обеспечить как ускорение поиска данных, так и уменьшение затрат на обновление индекса при вставках и удалениях. По этим причинам при решении задачи поиска в базах данных используют сбалансированные сильноветвящиеся деревья [14].

В данном случае сбалансированными деревьями называют такие деревья, что длины любых двух путей от корня до листьев одинаковы [15]. Сильноветвящимися же являются деревья, каждый узел которых ссылается на большое число потомков [16]. Эти условия обеспечивают минимальную высоту дерева для быстрого поиска и свободное пространство в узла для внесения изменений в базу данных без необходимости изменения индекса при каждой операции.

Наиболее используемыми деревьями поиска, имеющими описанные свойства, являются В-деревья и их разновидность — B^+ -деревья [14].

2.1.1 В-деревья

В-дерево — это сбалансированная, сильноветвящаяся древовидная, работающая с отсортированными значениями структура данных, операции вставки и удаления в которой не изменяют ее свойств [17]. Сбалансированность поддерживается путем построения структуры таким образом, что в узлах остаются положения для включения новых элементов [18]. Это осуществляется за счет свойств узлов, которые определяются порядком В-дерева m .

B-деревом порядка m [14; 18] называется дерево поиска, такое что:

- каждый узел имеет формат, описывающийся формулой 2.1:

$$(P_1, (K_1, Pr_1), P_2, (K_2, Pr_2), \dots, (K_{q-1}, K_{q-1}), P_q), \quad (2.1)$$

где $q \leq m$,

P_i — указатель на i -ого потомка в случае внутреннего узла или пустой указатель в случае внешнего (листа),

K_i — ключи поиска,

Pr_i — указатель на запись, соответствующую ключу поиска K_i ;

- для каждого узла выполняется $K_1 < K_2 < \dots < K_q$;
- для каждого ключа поиска X потомка, лежащего по указателю P_i выполняются условия, описывающиеся формулой 2.2:

$$\begin{aligned} K_{i-1} < X < K_i, & \text{ если } 1 < i < q, \\ X < K_i, & \text{ если } i = 1, \\ K_{i-1} < X, & \text{ если } i = q; \end{aligned} \quad (2.2)$$

- каждый узел содержит не более $m - 1$ ключей поиска или, что то же самое, имеет не более m потомков;
- каждый узел за исключением корня содержит не менее $\lceil m/2 \rceil - 1$ ключей поиска, или, что то же самое, имеет не менее $\lceil m/2 \rceil$ потомков;
- корень может содержать минимум один ключ, либо, что то же самое, иметь минимум два потомка;
- каждый узел за исключением листьев, содержащий $q - 1$ ключей, имеет q потомков;
- все листья находятся на одном и том же уровне.

В случае с индексами к каждому ключу поиска во всех узлах добавляется указатель на запись, соответствующую этому ключу. Другими словами, каждый узел содержит набор указателей, ссылающихся на дочерние узлы, и набор пар, каждая из которых состоит из ключа поиска и указателя, ссылающегося на данные. При этом записи с данными хранятся отдельно и частью *B*-дерева не являются [14].

Пример *B*-дерева представлен на рисунке 2.1.

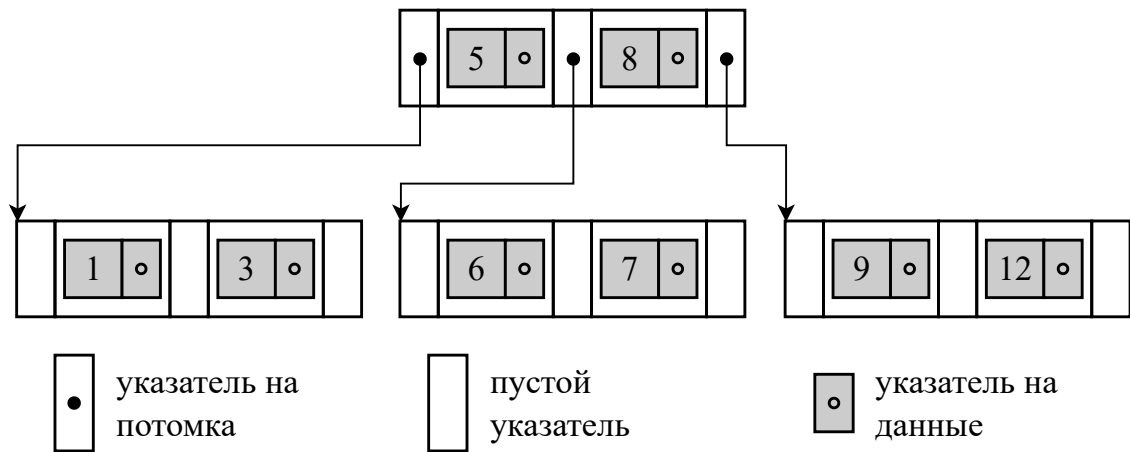


Рисунок 2.1 – Пример В-дерева

2.1.2 B^+ -деревья

Отличие от В-деревьев

Картинка

2.1.3 Обученные индексы

В-tree индексы можно рассматривать как модель сопоставления ключа позиции искомой записи в отсортированном массиве (рисунок 2.2).

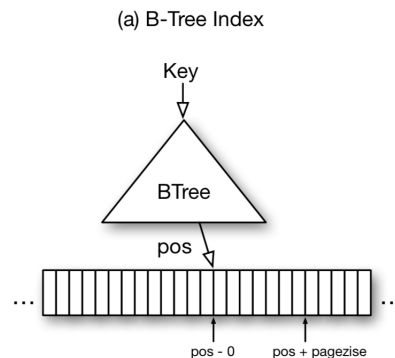


Рисунок 2.2 – В-деревья

Такие индексы как бы предсказывают положение записи с минимаксной ошибкой ($min_err = 0$, $max_err = page_size$). Поэтому можем заминить В-деревья на линейную модель также с минимаксной ошибкой (рисунок 2.3).

Так для предсказания можно представлять Range Index Models как модели функции распределения (рисунок 2.4):

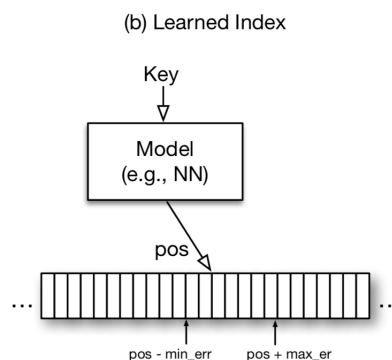


Рисунок 2.3 – Обученный индекс

$$\text{position} = F(\text{key}) \cdot N, \quad (2.3)$$

где $F(\text{key})$ — функция распределения, дающая оценку вероятности обнаружения ключа, меньшего или равного ключу поиска, то есть $P(X < \text{key})$;

N — количество ключей.

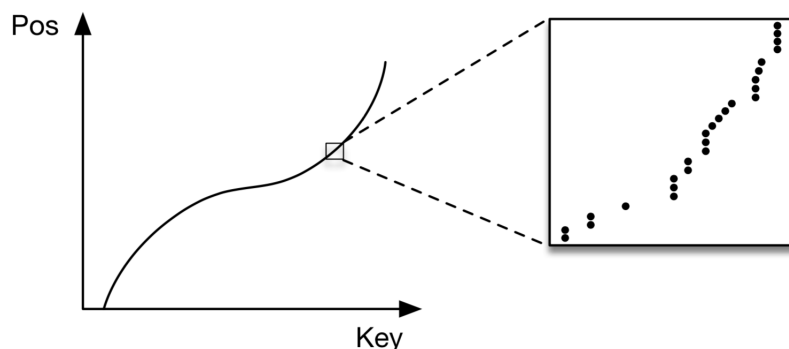


Рисунок 2.4 – Индекс как функция распределения

Можно построить индексы на основе рекурсивной модели (рисунок 2.5), в которой строится иерархия моделей из n уровней. Каждая модель на вход получает ключ, на основе которого выбирает модель на следующем уровне. Модели последнего этапа предсказывают положение записи.

Можно использовать различные модели: например, на верхнем использовать нейронные сети, а на нижних простые линейные регрессионные модели или даже простые В-деревья.

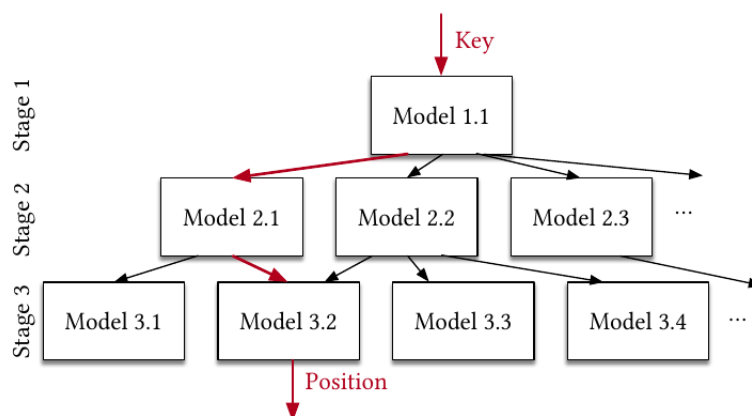


Рисунок 2.5 – Рекурсивная модель индекса

2.2 Индексы на основе хеш-таблиц

2.2.1 Хеш-индексы

2.2.2 Обученные хеш-индексы

Хеш-индексы можно рассматривать как модель сопоставления ключа позиции искомой записи в неупорядоченном массиве.

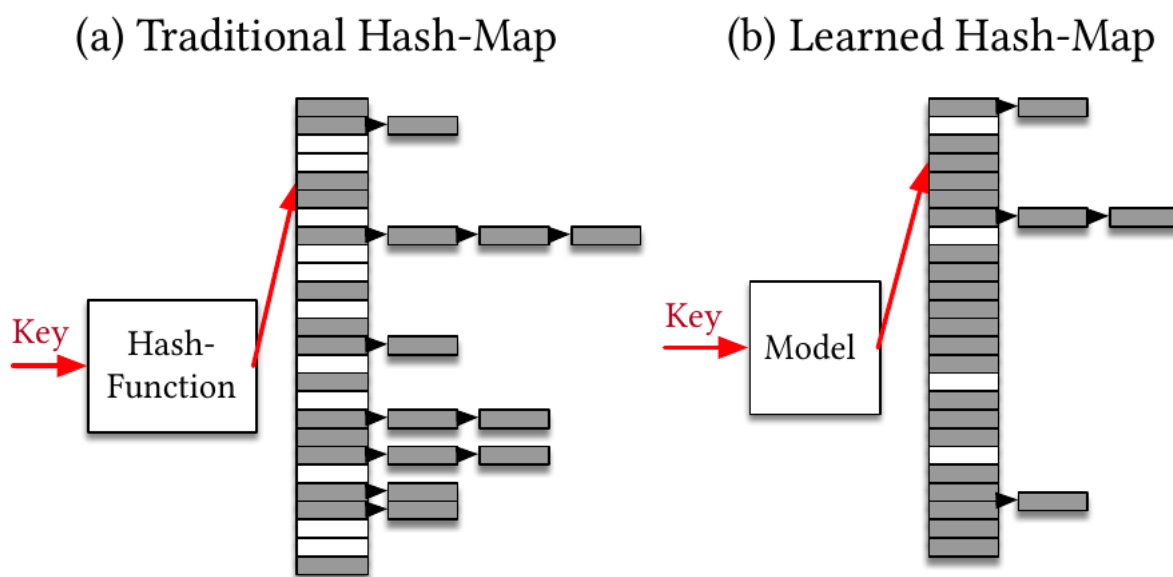


Рисунок 2.6 – Хеш-индексы

Функция распределения вероятностей распределения ключей один из возможных способов обучения хеш-индексов. Функция распределения масштабируется на размер хеш-таблицы M и для поиска положения записи аналогично случаю с В-деревьями используется формула:

$$h(K) = F(K) \cdot M, \quad (2.4)$$

где K — ключ.

2.3 Индексы на основе битовых карт

2.3.1 Фильтр Блума

2.3.2 Обученные индексы

Данные индексы можно рассматривать как модель проверки существования записи в массиве данных.

Фильтр Блума — алгоритм используемый для проверки существования записи.

Фильтр Блума использует массив бит размером m и k хеш-функций, каждая из которых сопоставляет ключ с одну из m позиций. Для добавления элемента в множество существующих значений ключ подается на вход каждой хеш-функции, возвращающих позицию бита, который должен быть установлен в единицу. Для проверки принадлежности ключа множеству, ключ также подается на вход k хеш-функций. Если какой-либо бит, соответствующий одной из возвращенных позиций, равен нулю, то ключ не входит во множество. Из этого следует, что данный алгоритм гарантирует отсутствие ложноотрицательных результатов.

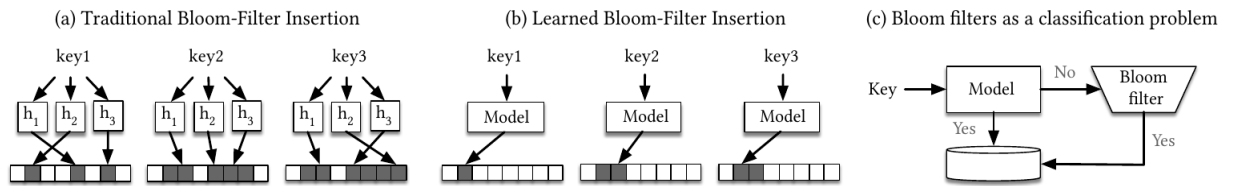


Рисунок 2.7 – Bitmap-индексы

В случае индексов существования необходимо обучить функцию таким образом, чтобы среди возвращенных значений для множества ключей были коллизии, аналогично для множества неключей, но при этом не было коллизий возвращенных значений для ключей и неключей.

В отличие от оригинального фильтра Блума, где $FNR = 0$, $FPR = const$, где $const$ выбрано априори, при обучении достигается заданное значение FPR

при $FNR = 0$ на реальных запросах.

3 Классификация существующих методов

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Исследование способов ускорения поисковых запросов в базах данных / Е. В. Коптенок [и др.] // Вестник образовательного консорциума средне-русский университет. Информационные технологии. — 2019. — N 1(13). — С. 24—27.
2. *Reinsel D., Gantz J., Rydning J.* The Digitization of the World From Edge to Core // IDC White Paper. — 2018.
3. *Носова Т. Н., Калугина О. Б.* Использование алгоритма битовых шкал для увеличения эффективности поисковых запросов, обрабатывающих данные с низкой избирательностью // Электротехнические системы и комплексы. — 2018. — N 1(38). — С. 63—67.
4. DAMA-DMBOK : Свод знаний по управлению данными. — 2-е изд. — М. : Олимп-Бизнес, 2020. — 828 с.
5. The Case for Learned Index Structures / T. Kraska [et al.] // Proceedings of the 2018 International Conference on Management of Data. — SIGMOD'18, June 10–15, 2018, Houston, TX, USA, 2018. — P. 489–504.
6. ALEX: An Updatable Adaptive Learned Index / J. Ding [et al.] // Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data. — 2020. — P. 969–984.
7. APEX: A High-Performance Learned Index on Persistent Memory (Extended Version) / B. Lu [et al.] // Proceedings of the VLDB Endowment. — 2022. — Vol. 15(3). — P. 597–610.
8. Updatable Learned Index with Precise Positions / J. Wu [et al.] // Proceedings of the VLDB Endowment. — 2021. — Vol. 14(8). — P. 1276–1288.
9. *Ferragina P., Vinciguerra G.* The PGM-index: a fully-dynamic compressed learned index with provable worst-case bounds // PVLDB. — 2020. — Vol. 13(8). — P. 1162–1175.
10. *Григорьев Ю. А., Плутенко А. Д., Плужникова О. Ю.* Реляционные базы данных и системы NoSQL: учебное пособие. — Благовещенск : Амурский гос. ун-т, 2018. — 424 с.

11. *Silberschatz A., Korth H. F., Sudarshan S.* Database System Concepts. — New York : McGraw-Hill, 2020. — 1344 p.
12. *Эдвард Сьоре.* Проектирование и реализация систем управления базами данных. — М. : ДМК Пресс, 2021. — 466 с.
13. *Осипов Д. Л.* Технологии проектирования баз данных. — М. : ДМК Пресс, 2019. — 498 с.
14. *Lemahieu W., Broucke S. vanden, Baesens B.* Principles of database management : the practical guide to storing, managing and analyzing big and small data. — Cambridge : Cambridge University Press, 2018. — 1843 p.
15. Encyclopedia of Database Systems / ed. by L. Liu, M. T. Özsu. — New York : Springer New York, 2018. — 4866 p.
16. *Mannino M. V.* Database Design, Application Development, and Administration. — Chicago : Chicago Business Press, 2019. — 873 p.
17. *Campbell L., Major C.* Database Reliability Engineering : Designing and Operating Resilient Database Systems. — Sebastopol : O'Reilly Media, 2018. — 560 p.
18. *Gupta G. K.* Database Management Systems. — Chennai : McGraw Hill Education (India) Private Limited, 2018. — 432 p.