

Многомерное Индексирование (Лекция 12)

Многомерное (и не только) индексирование

v5

Георгий Чернышев

Высшая Школа Экономики

chernishev@gmail.com

9 ноября 2020 г.

- ❶ Индексирование.
- ❷ Одномерное индексирование: B+-дерево.
- ❸ Многомерное индексирование
 - Введение
 - Запросы
 - Двухшаговая схема
 - R-дерево
- ❹ Методы разделения пространства на примере KD дерева
- ❺ Locality-Sensitive Hashing
- ❻ Машинное обучение и деревья

Способ ускорения доступа к данным, обычно какое-то упорядочивание.

- Используются древовидные структуры B^+ -tree, R -tree;
- Не всегда выгодно использовать: занимает место и может ухудшить общую производительность на обновлениях;
- Выбор атрибута (-ов) для индексирования отдается на откуп администратору СУБД;
- Существует много концептуально разных типов индексов.

B^+ -tree (1)

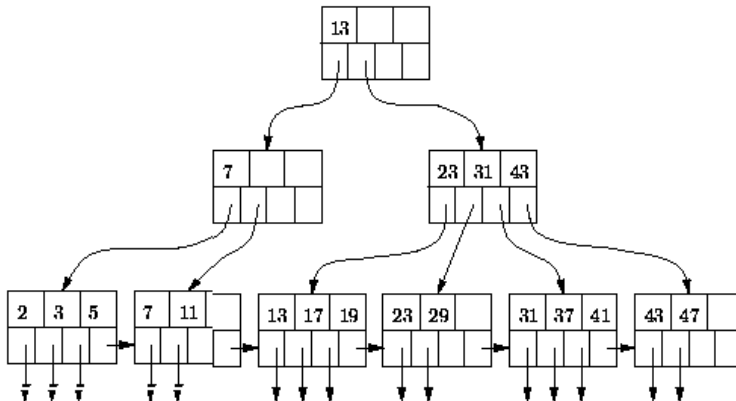
Это вариант B -tree, отличия:

- Все ключи в листьях;
- Листья прошиты для итерирования по данным;
- Все уровни кроме листового должны (хорошо было бы) чтобы помещались в оперативную память.

Используются во всех индустриальных (и большинстве игрушечных) СУБД.

B^+ -tree (2): пример

A B+ Tree



1

¹Изображение взято с <http://infolab.stanford.edu/~ullman/dbsi/win98/hw2.html>

- На самом деле она очень сложна!
- Надо думать о физическом представлении, тюнингу под кеш-память, параллельном доступе, восстановлении и прочей интеграции с другими подсистемами СУБД.
- Пример: достаточно игрушечный индекс на B^+ -tree это около 50 килобайт кода.

Индексирование N-мерном пространстве:

- Что индексируем:
 - Точки;
 - Объекты.
- Для чего индексируем:
 - СУБД;
 - GIS-системы (их, кажется, побольше будет).

Индекс разрабатывается под запросы [Manolopoulos et al., 2005]:

- Запрос на диапазон;
- Топологические запросы;

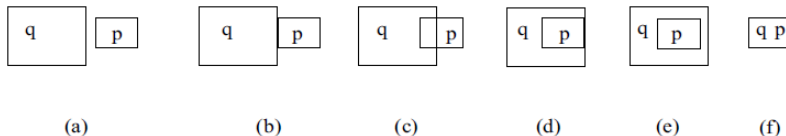


Fig. 4.3. Topological relations: (a) $\text{disjoint}(q, p)$; (b) $\text{meet}(q, p)$; (c) $\text{overlap}(q, p)$; (d) $\text{covers}(q, p)$; (e) $\text{contains}(q, p)$ or $\text{inside}(p, q)$; (f) $\text{equal}(q, p)$.

2

- Запросы на направление;
- Запросы к ближайшим соседям (разные варианты: прямые, обратные, условные и т.д.);
- Запросы с пространственным соединением: многопроходные, с пространственным предикатом.

²Изображение взято из [Manolopoulos et al., 2005]

Как индексировать объекты? Двухшаговая схема!

Описываем объекты с помощью MBR (min. bounding rectangle).

При поиске используем двухшаговую схему:

- Фильтрация, получение списка кандидатов;
- Проверка списка кандидатов, очистка от ложных срабатываний.

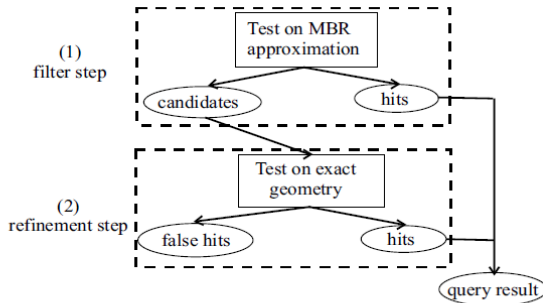


Fig. 4.2. Two-step query processing procedure.

R-tree (1)

- Древовидная структура для упрощения поиска;
- Можно сказать что это обобщение B^+ -tree на многомерный случай;
- Используется в индустриальных СУБД и GIS системах: PostgreSQL, Oracle, SQLite, PostGIS, MapInfo, ...
 - фактически индустриальный стандарт для многомерных данных малой размерности;
- Есть в Boost и куче других приложений, вне СУБД.

R-tree (2): популярность

Table 1.1. Access methods covered in this book, ordered by year of publication (continued).

Year	Access Method	Authors and References
1984	B-tree	Kamel, Fakirinas [105]
1984	B ⁺ -tree	Ng, Kanevich [101]
1984	TV-tree	Lin, Jagdish, Faloutsos [130]
1985	Q-tree	Maniopoulos, Nefriti, Papakonstantinou, Pratiiti [140]
1986	SR-tree	White, Jain [245]
1986	VAMSplit R-tree	White, Jain [244]
1986	X-tree	Wellisch, Klein, Kriegel [204]
1986	3D R-tree	Theodoridis, Vargianakis, Sellis [238]
1987	C-tree	Boussagoulas, Kettide [195]
1987	Linear Node Splitting	Ang, Tan [11]
1987	S-tree	Agarwal, Wolf, Wu, Eshedman [5]
1987	SB-tree	Kotemane, Sedh [158]
1987	STH R-tree	Leutenegger, Fajberg, Lopez [134]
1988	Bitonpoint R-tree	Kumar, Fortes, Edouard [120]
1988	HR-tree	Narciso, Silva [159, 156]
1988	Optimal Node Splitting	Chavira, Lopez, Leutenegger [73]
1988	RV-tree	Jorgensen, Lee [102]
1988	STLT	Chen, Choudhry, Rundenstein [42]
1988	TIS	Geurts, Lopez, Leutenegger [70]
1989	GIB	Choudhry, Chen, Rundenstein [45]
1989	B ⁺ -tree	Sellis, Jensen [201]
1989	2+1 B-tree	Narciso, Silva, Theodoridis [159]
1989	Branch Grabbing	Schick, Chen [200]
1989	Bitmap R-tree	Ang, Lee [12]
1989	TL-tree	Pfeifer, Johnson, Theodoridis [180]
1989	TPH-tree	Sellis, Jensen, Leutenegger, Lopez [202]
1989	AB-tree	Papadisa, Kintis, Zhang, An [178]
1989	Bio-tree	Agarwal, deBerg, Gudmundsson, Hammar, Haverkort [4]
1989	Compact B-tree	Huang, Lin, Lee [83]
1989	CB-tree	Kim, Cha, Kwon [110]
1989	Efficient B-tree	Lee, Papadisa [222]
1989	MV-tree	Lee, Papadisa [222]
1989	PE-tree	Kollins, Tzouros, Gimpelakis, Dello, Haldruphorris [113]
1989	RS-tree	Park, Lee, Kim [244]
1989	SOM-based R-tree	Oh, Fung, Katsikis, Makinouchi [162]
1989	TAB-tree	Protopar, Agarwal, Han, Pined, [92]
1989	ab-tree	Lee, Papadisa, Zhang, [221]
1989	Buffer B-tree	Argo, Hartzel, Valenzuela, Vitter, [66]
1989	Q-tree	Theodoridis, Peter, Theodoridis, [52]
1989	DR-tree	Lee, Ching, [135]
1989	HBM R-tree	Jin, Jagdish, [104]
1989	Lazy Update R-tree	Kwon, Lee, Lee, [124]
1989	Low Stalling Number	deBerg, Hammar, Overmars, Gudmundsson, [66]
1989	VLT R-tree	Protopar, Xu, Kalashnikov, Arif, Hammar, [181]
1989	PRR-tree	Pentacos, [87]
1989	LR-tree	Bennett, Natsopoulos, Manolopoulos, [11]
1989	OMT R-tree	Lee, Lee, [131]
1989	Partitioned R-tree	Bennett, Natsopoulos, Manolopoulos, [11]
1989	Q-tree	Xie, Protopar, [248]
1989	Seeded Clustering	Lee, Moon, Lee, [132]
1989	STLT	Chakka, Brounough, Patel, [38]
1989	TPH-tree	Lee, Papadisa, Shi, [221]
1989	TR-tree	Park, Lee, [185]
1989	Merging B-trees	Vassiliou, Natsopoulos, Bennett, [240]
1989	MON-tree	Almeida, Guting, [7]
1989	PR-tree	Argo, deBerg, Hammar, [11, 135]
1989	R-tree	Pelade, Sabaria, Jensen, [138]
1989	V-MAT	Gorawski, Makris, [73, 74]

Table 1.1. Access methods covered in this book, ordered by year of publication.

Year	Access Method	Authors and References
1984	B-tree	Gottman [21]
1985	Packed B-tree	Boussagoulas, Gauthier [199]
1987	B ⁺ -tree	Sellis, Boussagoulas, Faloutsos [211]
1989	Cell-tree	Grossman [77]
1989	F-tree	Jagdish, Wei and 1981 Scheiela [204]
1989	B ⁺ -tree	Berkman, Kriegel, Schneider, Seeger [19]
1989	B ⁺ -tree	An, Han, Lu [249]
1989	Sphere-tree	Osterman [161]
1989	Independent B-tree	Kamel, Fakirinas [103]
1989	MV R-tree	Kamel, Fakirinas [103]
1989	Supertree B-tree	Kamel, Fakirinas [103]
1989	Efficient Packed B-tree	Kamel, Fakirinas [104]

4

R-tree (3): определение

Согласно [Manolopoulos et al., 2005] R-дерево — это древовидная структура данных, заданная парой (m, M) со следующими свойствами:

- Каждый лист может содержать до M записей, минимально $2 \leq m \leq M/2$.
- Каждая запись в листе представлена в форме (mbr, oid) , где mbr это минимальный ограничивающий прямоугольник, а oid — идентификатор объекта.
- Количество записей хранящихся во внутреннем узле также должно принадлежать $[m; M]$. Каждая запись в узле представляет собой пару (mbr, p) , где p — указатель на ребенка узла, а mbr содержит в себе mbr ребенка.
- Дерево сбалансировано — все листья находятся на одном уровне.

R-tree (4): пример

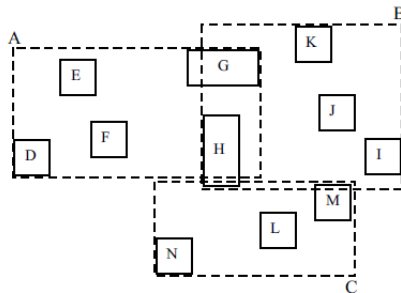


Fig. 1.2. An example of data MBRs and their MBRs.

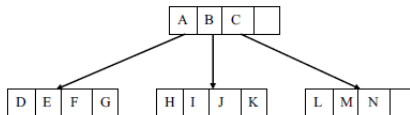


Fig. 1.3. The corresponding R-tree.

5

⁵ Изображение взято из [Manolopoulos et al., 2005]

- Теперь у нас не диапазоны, а “коробочки” — многомерные прямоугольники;
- Сложно расщеплять вершину: NP -трудная задача + много критериев;
- Сложно искать, MBR могут пересекаться \rightarrow надо проверить много путей до листьев;

```
Algorithm RangeSearch(TypeNode  $RN$ , TypeRegion  $Q$ )
/* Finds all rectangles that are stored in an R-tree with root node  $RN$ , which are
intersected by a query rectangle  $Q$ . Answers are stored in the set  $\mathcal{A}$  */

1.  if  $RN$  is not a leaf node
2.      examine each entry  $e$  of  $RN$  to find those  $e.mbr$  that intersect  $Q$ 
3.      foreach such entry  $e$  call RangeSearch( $e.ptr, Q$ )
4.  else //  $RN$  is a leaf node
5.      examine all entries  $e$  and find those for which  $e.mbr$  intersects  $Q$ 
6.      add these entries to the answer set  $\mathcal{A}$ 
7.  endif
```

Fig. 1.4. The R-tree range search algorithm.

Algorithm Insert(TypeEntry E , TypeNode RN)

/* Inserts a new entry E in an R -tree with root node RN */

1. Traverse the tree from root RN to the appropriate leaf. At each level, select the node, L , whose MBR will require the minimum area enlargement to cover $E.mbr$
2. In case of ties, select the node whose MBR has the minimum area
3. if the selected leaf L can accommodate E
4. Insert E into L
5. Update all MBRs in the path from the root to L , so that all of them cover $E.mbr$
6. else // L is already full
7. Let \mathcal{E} be the set consisting of all L 's entries and the new entry E
 Select as seeds two entries $e_1, e_2 \in \mathcal{E}$, where the distance between e_1 and e_2 is the maximum among all other pairs of entries from \mathcal{E}
 Form two nodes, L_1 and L_2 , where the first contains e_1 and the second e_2
8. Examine the remaining members of \mathcal{E} one by one and assign them to L_1 or L_2 , depending on which of the MBRs of these nodes will require the minimum area enlargement so as to cover this entry
9. if a tie occurs


```
10.     Assign the entry to the node whose MBR has the smaller area
11. endif
12. if a tie occurs again
13.     Assign the entry to the node that contains the smaller number of entries
14. endif
15. if during the assignment of entries, there remain  $\lambda$  entries to be assigned
    and the one node contains  $m - \lambda$  entries
16.     Assign all the remaining entries to this node without considering
        the aforementioned criteria
        /* so that the node will contain at least  $m$  entries */
17. endif
18. Update the MBRs of nodes that are in the path from root to  $L$ , so as to
    cover  $L_1$  and accommodate  $L_2$ 
19. Perform splits at the upper levels if necessary
20. In case the root has to be split, create a new root
21. Increase the height of the tree by one
22. endif
```

Fig. 1.5. The R -tree insertion algorithm.

Задача расщепления вершины

Строки 6–17 на самом деле отдельный алгоритм.

- Много критериев: суммарная площадь, периметр, пересечение и т.д.

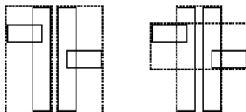


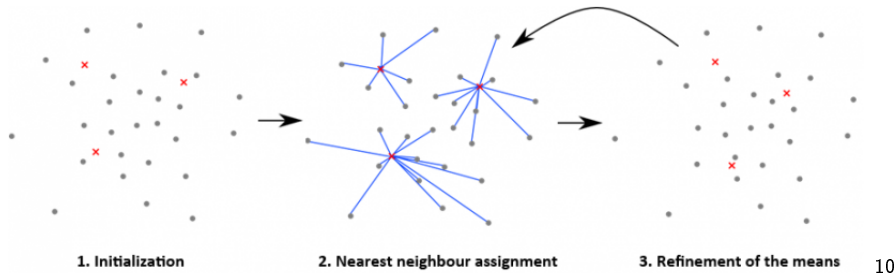
Fig. 1.6. Left: bad split; Right: good split.

9

- NP -трудная задача, доказано;
- А значит и много подходов к решению: Al-Badarneh's et al. split, Corner-Based split, K-means split, Hilbert split, RR^* split, R^* split, Greenes split, Guttman Quadratic and Linear splits, Ang and Tan split, Double-Sorting Based split.

⁹ Изображение взято из [Manolopoulos et al., 2005]

Применение K-means



- Исходная работа [Brakatsoulas et al., 2002]
- Проблема — для сходимости в случае прямоугольников нужно либо доказывать ее, либо нужны:
 - 1 правильная метрика расстояния между многоугольниками;
 - 2 правильная формула центраида.
- Математически обоснованное решение [Grigorev and Chernishev, 2016];

Основные варианты [Paradopoulos et al., 2009]:

- R^* — повторная вставка при расщеплении;
- R^+ — объект может покрываться несколькими mbr;
- Гильбертово R -дерево — учет точки на гильбертовой кривой;
- RR^* — самый сильный вариант, много сложных эвристик;
- Всего есть более 70 вариантов;

Что бывает еще?

Классификация:

- Методы выделения пространства — *R-tree*, *RD-tree*, *SR-tree*;
- Методы разделения пространства — *quadtree*, *Kd-tree*, *tries*;

Реализация в постгрес (обобщенная модель с транзакционностью):

- GiST¹¹;
- SP-GiST¹² [Eltabakh et al., 2006];

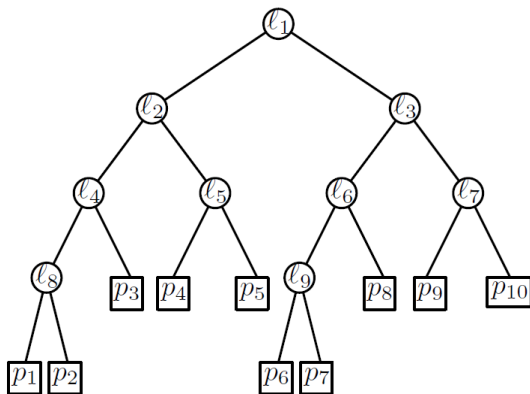
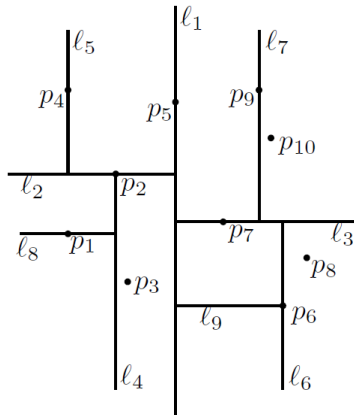
¹¹<http://www.sai.msu.su/~megera/postgres/gist/>

¹²<https://www.postgresql.org/docs/9.2/static/spgist-intro.html>

Суть:

- Имея n -мерный набор точек расщепляем его последовательно по $x_1, x_2, \dots, x_n, x_1, x_2, \dots$;
- На i -м шаге расщепления делим наш набор по медиане i -ой координаты на две части, “проводим” черту разбивающую наше пространство на две части;
- На $i + 1$ -м шаге аналогично поступаем с каждой половиной точек оставшейся от i -го.

Двумерный пример



13

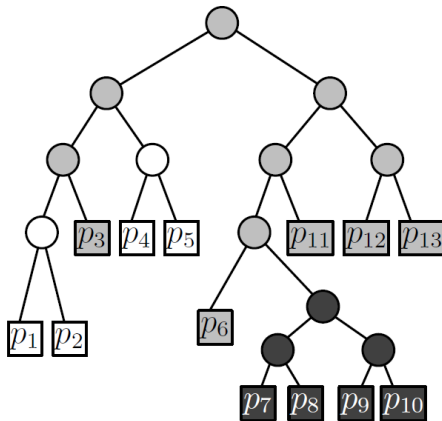
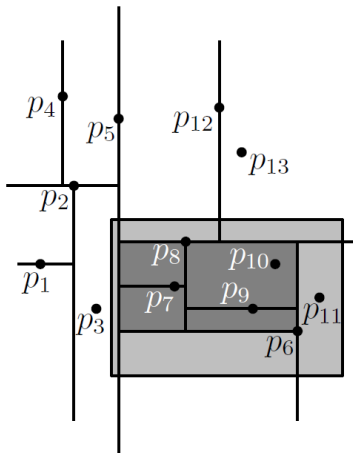
Algorithm BUILDKDTREE($P, depth$)

1. **if** P contains only one point
2. **then return** a leaf storing this point
3. **else if** $depth$ is even
4. **then** Split P with a vertical line ℓ through the median x -coordinate into P_1 (left of or on ℓ) and P_2 (right of ℓ)
5. **else** Split P with a horizontal line ℓ through the median y -coordinate into P_1 (below or on ℓ) and P_2 (above ℓ)
6. $v_{\text{left}} \leftarrow \text{BUILDKDTREE}(P_1, depth + 1)$
7. $v_{\text{right}} \leftarrow \text{BUILDKDTREE}(P_2, depth + 1)$
8. Create a node v storing ℓ , make v_{left} the left child of v , and make v_{right} the right child of v .
9. **return** v

14

Запросы

Регион либо хранится с каждым узлом, либо вычисляется налету.



15

Как на практике обстоят дела с деревьями?

- С ростом размерности очень сильно растет время построения и время исполнения запросов.
- Что происходит? Приходится проверять всё больший и больший % индекса — **curse of dimensionality**;
- Например, в 8-ми мерном пространстве поиск может быть в 16 раз медленнее чем в двумерном на индексе с R -tree;
- Понижение размерности — PCA, SVD и прочее — не помогает;
- **Математики доказали что это неизбежно для деревьев**;

Поэтому, область применения деревьев — R -tree до 10 измерений, отдельные варианты до 30-40 (RR^* -tree). В тоже время, индустрии нужны десятки-сотни тысяч...

Что делать?

Locality-Sensitive Hashing (LSH) [Gionis et al., 1999], суть:

- Имеем набор хеш-функций, они распределяют объекты по ведрам;
- Хеш-функции заданы так, чтобы вероятность коллизии была максимальна если объекты близки.

Увы:

- Работает только для kNN, дубликатов и еще некоторых типов;
- Не точно: не только возвращает то, что не годится, но вдобавок еще и “теряет” нужные данные!
- Впрочем, есть оценки насколько неточно.

Вроде как, работает на практике.

Что еще бывает из деревьев?

- M-tree [Ciaccia et al., 1997] — дерево для запросов по подобию с произвольной метрикой;
- ND-tree [Qian et al., 2006] — дерево для индексирования неупорядоченных дискретных многомерных значений, например генетических последовательностей.
- VP-tree [Yianilos, 1993] — в метрическом пространстве выбираем точку и делим данные на те, что ближе трешолда и те что дальше.
- ...

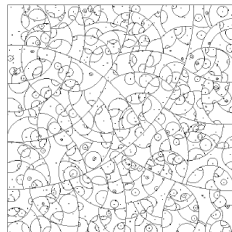


Figure 1: vp-tree decomposition

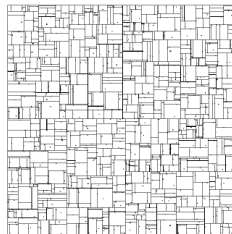


Figure 2: kd-tree decomposition

16

Кто хочет еще больше деревьев — смотрите книжку Hanan Samet'a.

Кажется с перспективами деревьев всё довольно плохо, есть вероятность что в ближайшие годы они очень сильно потеряют в значимости. А может даже будут просто убраны из вводного курса информатики!

Что случилось? Машинное обучение наступает :(

J. Ding et al. 2020. ALEX: An Updatable Adaptive Learned Index. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20). Association for Computing Machinery, New York, NY, USA, 969–984.

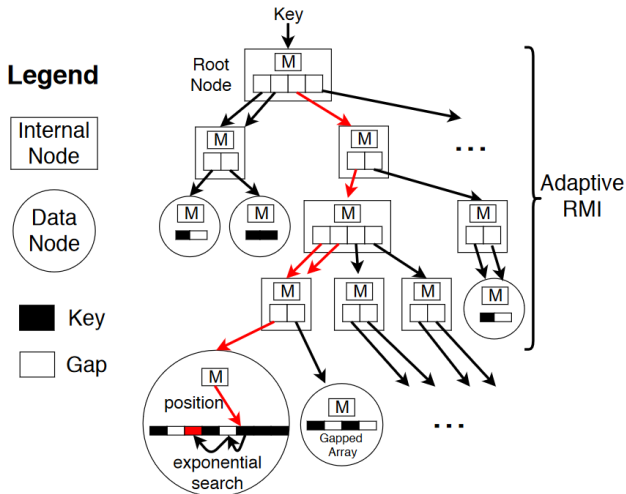


Figure 2: ALEX Design

ALEX: бенчмарк

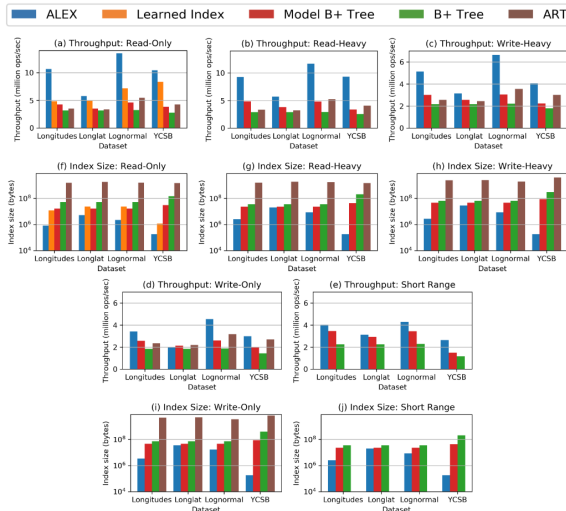


Figure 9: ALEX vs. Baselines: Throughput & Index Size.
Throughput includes model retraining time.

18

ALEX: бенчмарк II

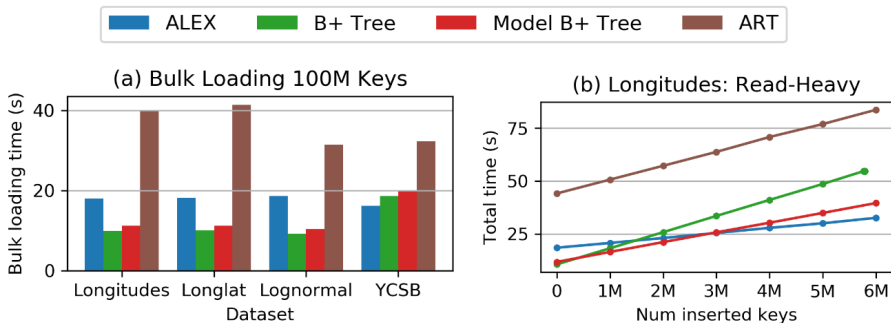
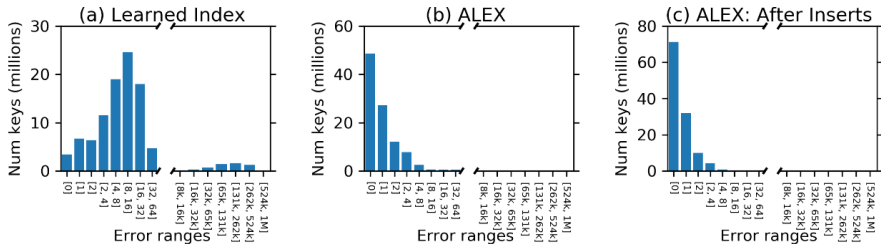


Figure 11: ALEX takes 50% more than time than B+ Tree to bulk load on average, but quickly makes up for this by having higher throughput.

19



А что с многомерностью?

Z. Yang et al. 2020. Qd-tree: Learning Data Layouts for Big Data Analytics. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20). Association for Computing Machinery, New York, NY, USA, 193–208.



Peter N. Yianilos. 1993. Data structures and algorithms for nearest neighbor search in general metric spaces. In Proceedings of the fourth annual ACM-SIAM symposium on Discrete algorithms (SODA '93). Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 311-321.



Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99), Malcolm P. Atkinson, Maria E. Orlowska, Patrick Valduriez, Stanley B. Zdonik, and Michael L. Brodie (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 518–529.



Paolo Ciaccia, Marco Patella, and Pavel Zezula. 1997. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB '97), Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 426–435.



Gang Qian, Qiang Zhu, Qiang Xue, and Sakti Pramanik. 2006. Dynamic indexing for multidimensional non-ordered discrete data spaces using a data-partitioning approach. *ACM Trans. Database Syst.* 31, 2 (June 2006), 439–484.
DOI=<http://dx.doi.org/10.1145/1138394.1138395>



M. Y. Eltabakh, R. Eltarras and W. G. Aref, "Space-Partitioning Trees in PostgreSQL: Realization and Performance," 22nd International Conference on Data Engineering (ICDE'06), 2006, pp. 100–100. doi: 10.1109/ICDE.2006.146



Apostolos N. Papadopoulos, Antonio Corral, Alexandros Nanopoulos, Yannis Theodoridis. R-Tree (and Family). Ling Liu, M. Tamer Özsu (Eds.): *Encyclopedia of Database Systems*. 2453–2459. Springer US 2009, ISBN 978-0-387-35544-3, 978-0-387-39940-9.



Sotiris Brakatsoulas, Dieter Pfoser, and Yannis Theodoridis. 2002. Revisiting R-Tree Construction Principles. In *Proceedings of the 6th East European Conference on Advances in Databases and Information Systems (ADBIS '02)*, Yannis Manolopoulos and Pavol Návrát (Eds.). Springer-Verlag, London, UK, UK, 149–162.



Valentin Grigorev and George Chernishev. 2016. K-means Split Revisited: Well-grounded Approach and Experimental Evaluation. In Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16). ACM, New York, NY, USA, 2251–2252. DOI: <http://dx.doi.org/10.1145/2882903.2914833>



Yannis Manolopoulos, Alexandros Nanopoulos, Apostolos N. Papadopoulos, and Yannis Theodoridis. 2005. R-Trees: Theory and Applications. Springer Publishing Company, Incorporated.