Многомерное Индексирование (Лекция 12)

Многомерное (и не только) индексирование

v5

Георгий Чернышев

Высшая Школа Экономики

chernishev@gmail.com

9 ноября 2020 г.

План

1. Индексирование.
2. Одномерное индексирование: B+-дерево.
3. Многомерное индексирование
   - Введение
   - Запросы
   - Двухшаговая схема
   - R-дерево
4. Методы разделения пространства на примере KD дерева
5. Locality-Sensitive Hashing
6. Машинное обучение и деревья

Индекс

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.

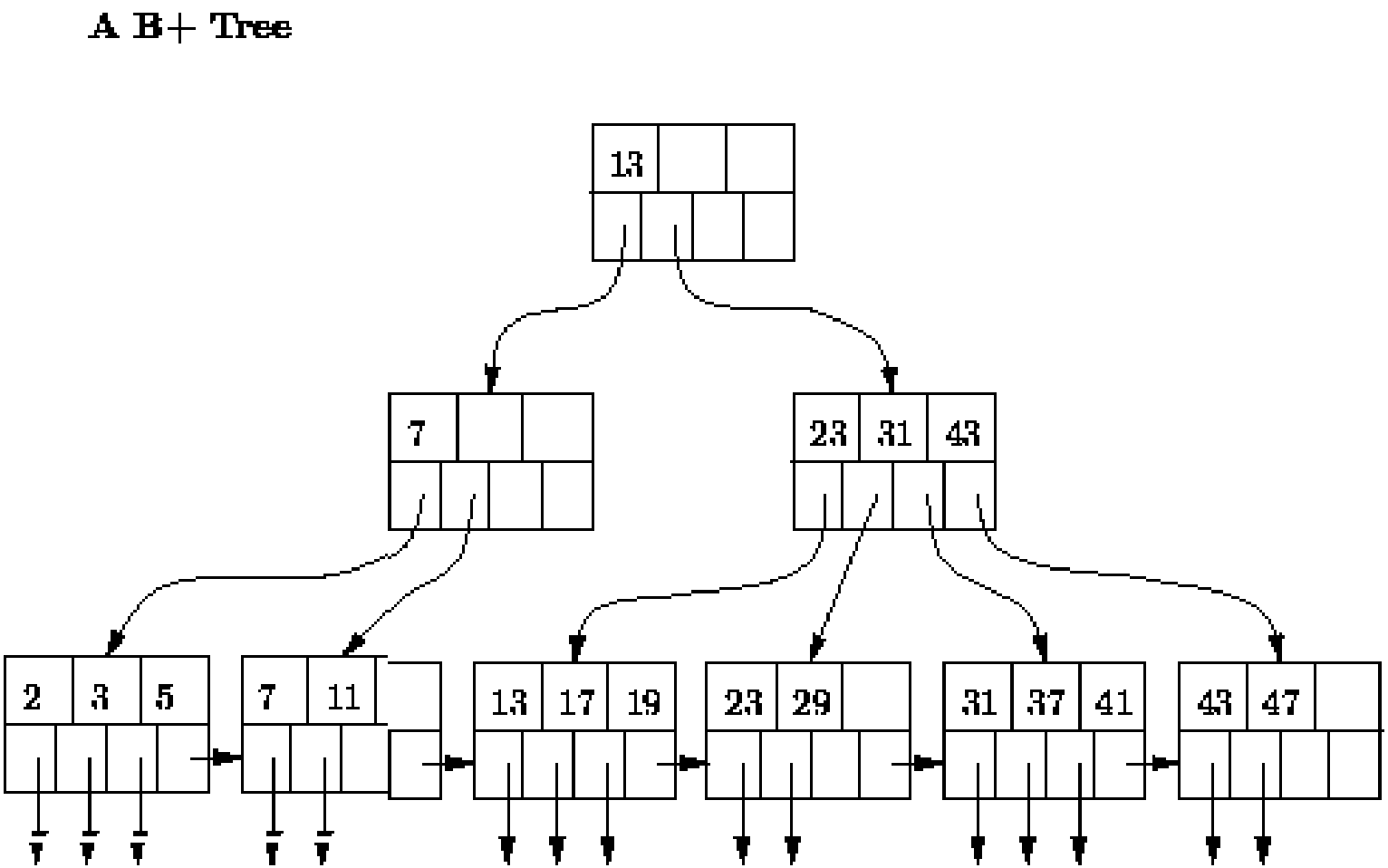
B^+-tree (1)

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



1

[1] Изображение взято с http://infolab.stanford.edu/ ullman/dbsi/win98/hw2.html

О реализации

- 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.

Многомерное индексирование

Индексирование N-мерном пространстве:

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

Запросы

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

- Запрос на диапазон;
- Топологические запросы;

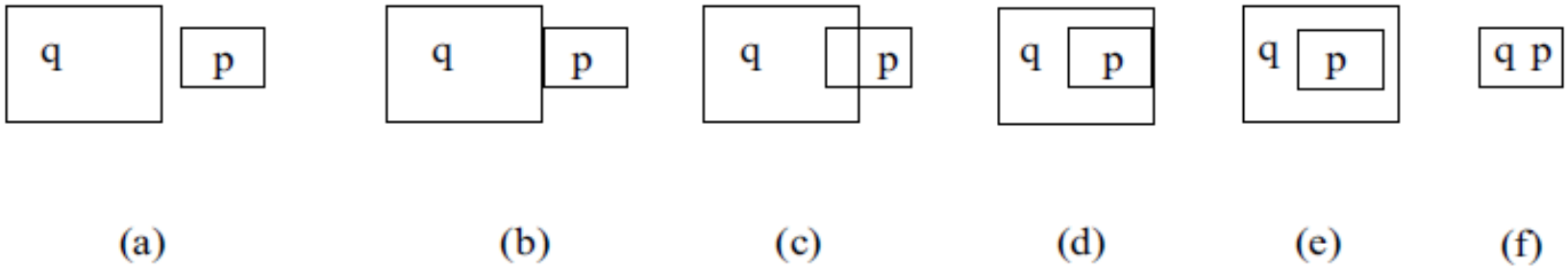


Fig. 4.3. Topological relations: (a) disjoint(q, p); (b) meet(q, p); (c) overlap(q, p); (d) covers(q, p); (e) contains(q, p) or inside(p, q); (f) equal(q, p).

2

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

[2] Изображение 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.

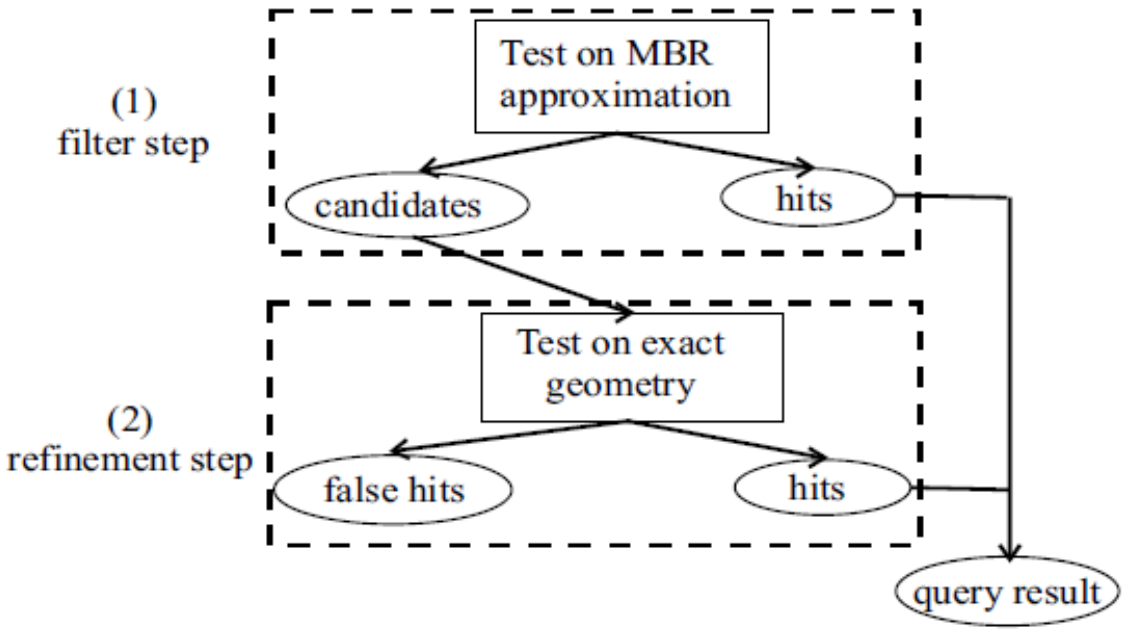


Fig. 4.2. Two-step query processing procedure.

[3]

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

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
1994	Hilbert R-tree	Kamel, Faloutsos [105]
1994	R-link	Ng, Kameda [161]
1994	TV-tree	Lin, Jagadish, Faloutsos [138]
1996	QR-tree	Manolopoulos, Nardelli, Papadopoulos, Proietti [146]
1996	SS-tree	White, Jain [245]
1996	VAMSplit R-tree	White, Jain [244]
1996	X-tree	Berchtold, Keim, Kriegel [24]
1996	3D R-tree	Theodoridis, Vazirgiannis, Sellis [238]
1997	Cubtree	Roussopoulos, Kotidis [198]
1997	Linear Node Splitting	Ang, Tan [11]
1997	S-tree	Aggrawal, Wolf, Wu, Epelman [5]
1997	SR-tree	Katayama, Satoh [108]
1997	STR R-tree	Leutenegger, Edgington, Lopez [134]
1998	Bitemporal R-tree	Kumar, Tsotras, Faloutsos [125]
1998	HR-tree	Nascimento, Silva [158, 159]
1998	Optimal Node Splitting	Garcia, Lopez, Leutenegger [71]
1998	R^*_S-tree	Juergens, Lenz [102]
1998	STLT	Chen, Choubey, Rundensteiner [42]
1998	TGS	Garcia, Lopez, Leutenegger [70]
1999	GBI	Choubey, Chen, Rundensteiner [47]
1999	R^{ST}-tree	Saltenis, Jensen [201]
1999	2+3 R-tree	Nascimento, Silva, Theodoridis [159]
2000	Branch Grafting	Schrek, Chen [208]
2000	Bitmap R-tree	Ang, Tan [12]
2000	TB-tree	Pfoser, Jensen, Theodoridis [189]
2000	TPR-tree	Saltenis, Jensen, Leutenegger, Lopez [202]
2001	aR-tree	Papadias, Kanlis, Zhang, Tao [170]
2001	Box-tree	Agarwal, deBerg, Gudmundsson, Hammar, Haverkort [4]
2001	Compact R-tree	Huang, Lin, Lin [93]
2001	CR-tree	Kim, Cha, Kwon [110]
2001	Efficient HR-tree	Tao, Papadias [222]
2001	MV3R-tree	Tao, Papadias [223]
2001	PPR-tree	Kollios, Tsotras, Gunopulos, Delis, Hadjieleftheriou [113]
2001	RS-tree	Park, Heu, Kim [184]
2001	SOM-based R-tree	Oh, Feng, Kaneko, Makinouchi [162]
2001	STAR-tree	Procopiuc, Agarwal, Har-Peled, [192]
2002	aP-tree	Tao, Papadias, Zhang, [228]
2002	Buffer R-tree	Arge, Hinrichs, Vahrenhold, Vitter, [16]
2002	cR-tree	Brakatsoulas, Pfoser, Theodoridis, [32]
2002	DR-tree	Lee, Chung, [133]
2002	HMM R-tree	Jin, Jagadish, [100]
2002	Lazy Update R-tree	Kwon, Lee, Lee, [127]
2002	Low Stabbing Number	deBerg, Hammar, Overmars, Gudmundsson, [56]
2002	VCI R-tree	Prabhakar, Xia, Kalashnikov, Aref, Hambrusch, [191]
2003	FNR-tree	Frentzos, [67]
2003	LR-tree	Bozanis, Nanopoulos, Manolopoulos, [31]
2003	OMT R-tree	Lee, Lee, [131]
2003	Partitioned R-tree	Bozanis, Nanopoulos, Manolopoulos, [31]
2003	Q+R-tree	Xia, Prabhakar, [248]
2003	Seeded Clustering	Lee, Moon, Lee, [132]
2003	SETI	Chakka, Everspaugh, Patel, [38]
2003	TPR^*-tree	Tao, Papadias, Sun, [227]
2003	TR-tree	Park, Lee, [185]
2004	Merging R-trees	Vasatitis, Nanopoulos, Bozanis, [240]
2004	MON-tree	Almeida, Guting, [7]
2004	PR-tree	Arge, deBerg, Haverkort, Yi, [15]
2004	R^{PPF}-tree	Pelanis, Saltenis, Jensen, [188]
2004	VMAT	Gorawski, Malczok, [73, 74]

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

Year	Access Method	Authors and References
1984	R-tree	Guttman [81]
1985	Packed R-tree	Roussopoulos, Leifker [199]
1987	R^+-tree	Sellis, Roussopoloulos, Faloutsos [211]
1989	Cell-tree	Guenther [77]
1990	P-tree	Jagadish, [96] (and 1993 Schiwietz [206])
1990	R^*-tree	Beckmann, Kriegel, Schneider, Seeger [19]
1990	RT-tree	Xu, Han, Lu [249]
1990	Sphere-tree	Oosterom [164]
1992	Independent R-trees	Kamel, Faloutsos [103]
1992	MX R-tree	Kamel, Faloutsos [103]
1992	Supernode R-tree	Kamel, Faloutsos [103]
1993	Hilbert Packed R-tree	Kamel, Faloutsos [104]

4

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

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 при- надлежать $[m; M]$. 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

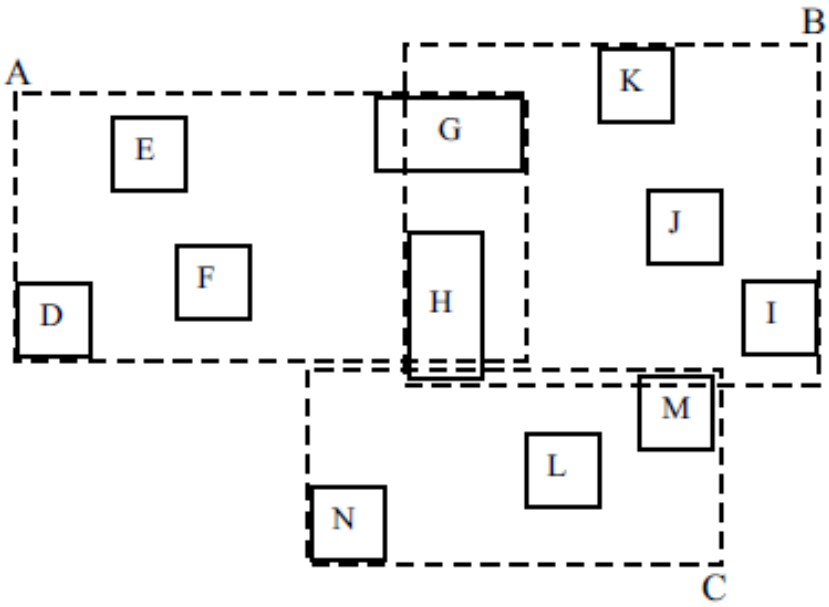


Fig. 1.2. An example of data MBRs and their MBRs.

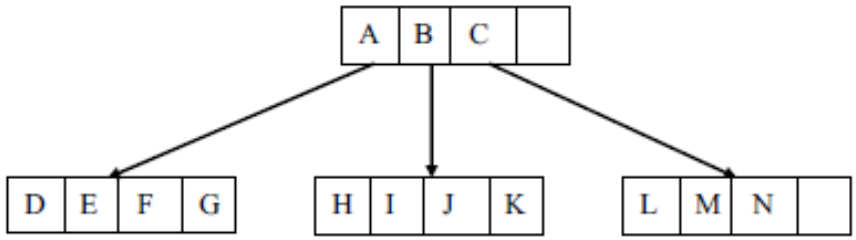


Fig. 1.3. The corresponding R-tree.

[5]

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

R-tree (5): свойства

- 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;

Вычисление диапазонного запроса

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 A
7.  endif
```

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

6

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

Вставка в R-tree I

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

7

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

Вставка в R-tree II

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 λ 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.

8

Задача расщепления вершины

Строки 6–17 на самом деле отдельный алгоритм.

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

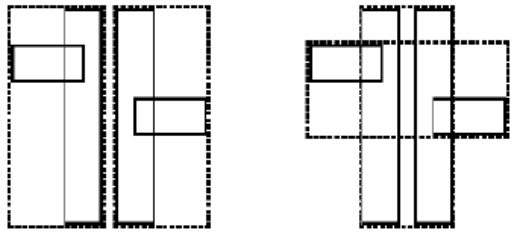


Fig. 1.6. Left: bad split; Right: good split.

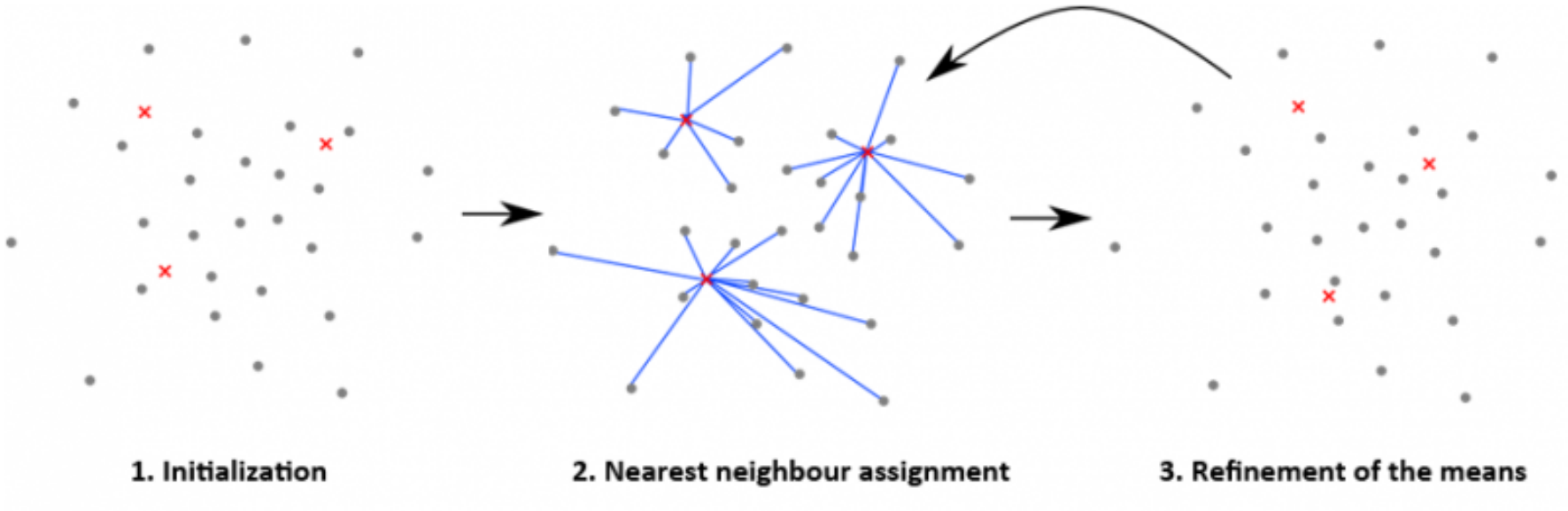
9

- *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



10

- 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 центроида.
- Математически обоснованное решение [Grigorev and Chernishev, 2016];

[10] Изображение взято из https://prateekvjoshi.com/2013/06/06/what-is-k-means-clustering/

R-tree: варианты

Основные варианты [Papadopoulos et al., 2009]:

- R^* — повторная вставка при расщеплении;
- R^+ — объект может покрываться несколькими mbr;
- Гильбертово R-дерево — учет точки на гильбертовой кривой;
- 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):

- GiST[11];
- SP-GiST[12] [Eltabakh et al., 2006];

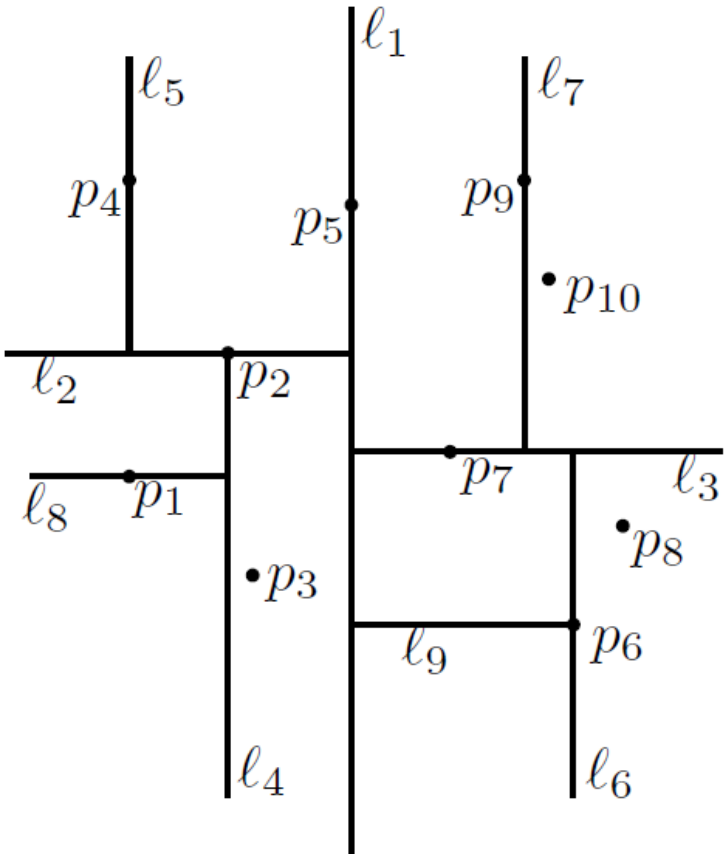
[11] http://www.sai.msu.su/ megera/postgres/gist/
[12] https://www.postgresql.org/docs/9.2/static/spgist-intro.html

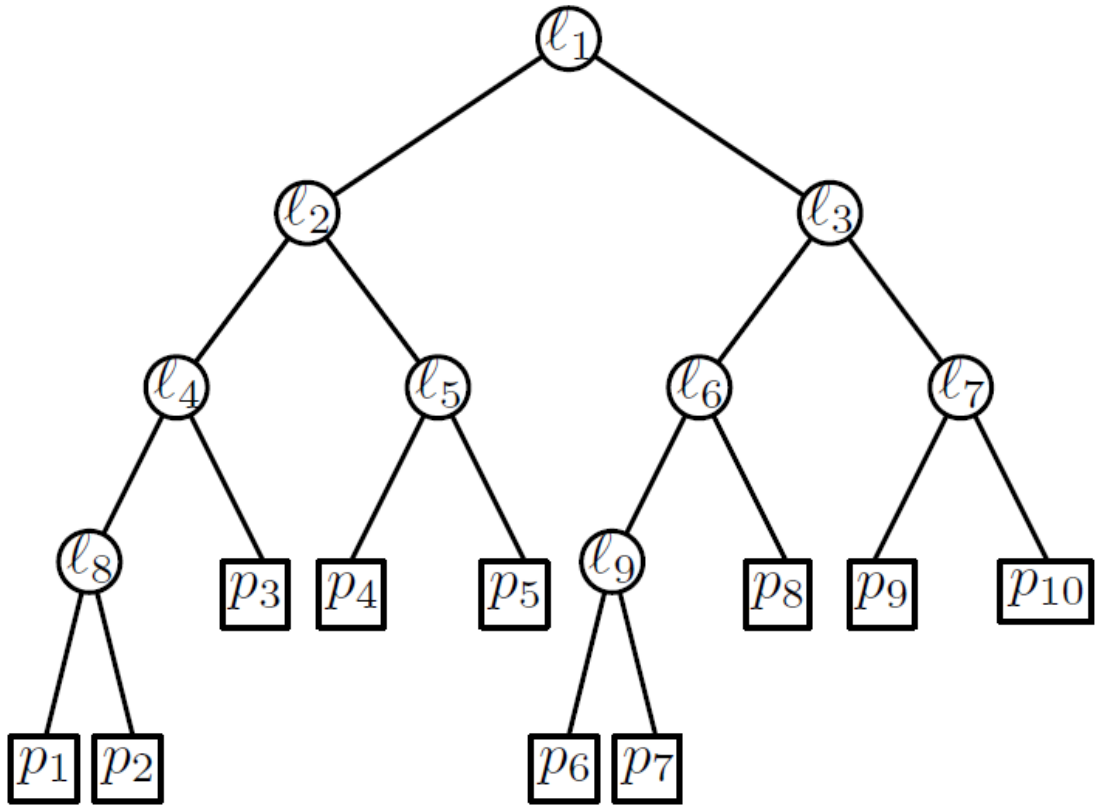
Kd-дерево

Суть:

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

Двумерный пример





13

[13] Изображение взято из https://www.cise.ufl.edu/class/cot5520fa09/CG_RangeKDtrees.pdf

Как строить

Algorithm BUILDKDTREE($P, depth$)

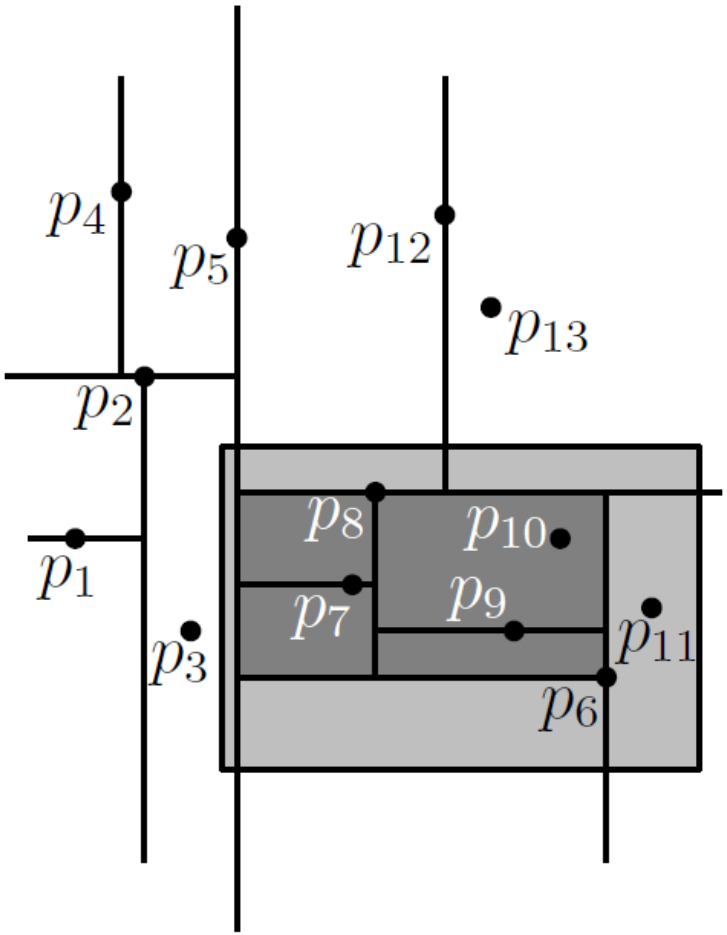
1. **if** P contains only one point
2. **then return** a leaf storing this point
3. **else if** $depth$ is even
4. **then** Split P with a vertical line ℓ through the median x-coordinate into P_1 (left of or on ℓ) and P_2 (right of ℓ)
5. **else** Split P with a horizontal line ℓ through the median y-coordinate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

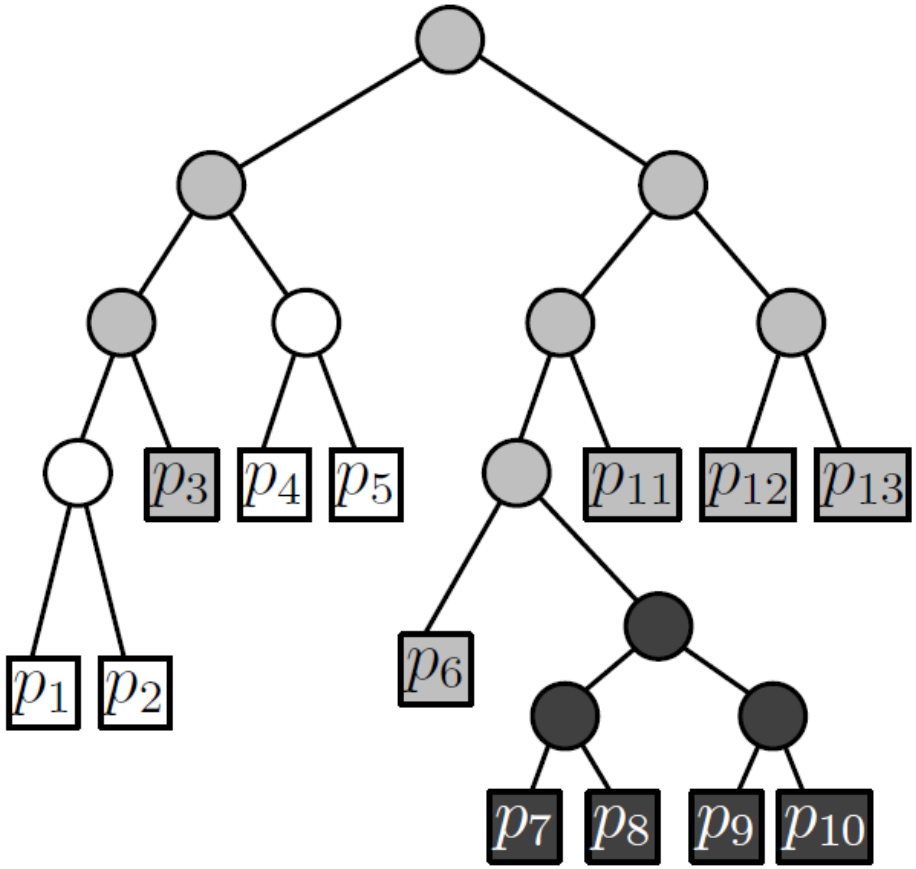
14

[14] Изображение взято из https://www.cise.ufl.edu/class/cot5520fa09/CG_RangeKDtrees.pdf

Запросы

Регион либо хранится с каждым узлом, либо вычисляется налету.





15

[15] Изображение взято из https://www.cise.ufl.edu/class/cot5520fa09/CG_RangeKDtrees.pdf

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 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.
- ...

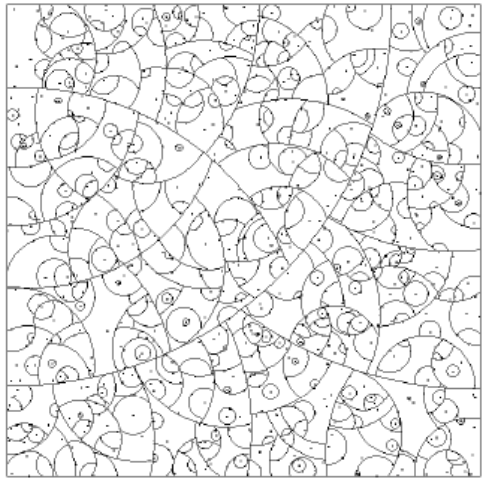


Figure 1: vp-tree decomposition

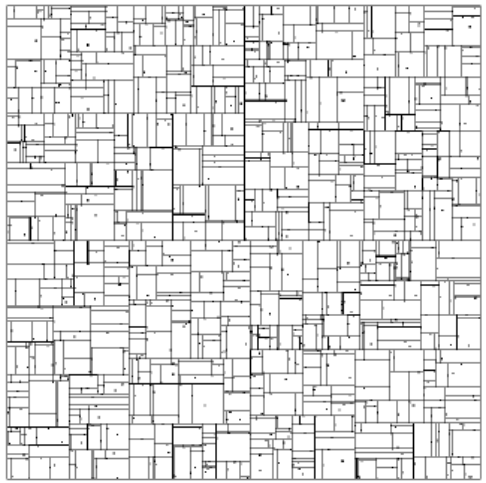


Figure 2: kd-tree decomposition [16]

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

[16] Изображение взято из [Yianilos, 1993]

Деревья в 202* году

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.

ALEX: идея

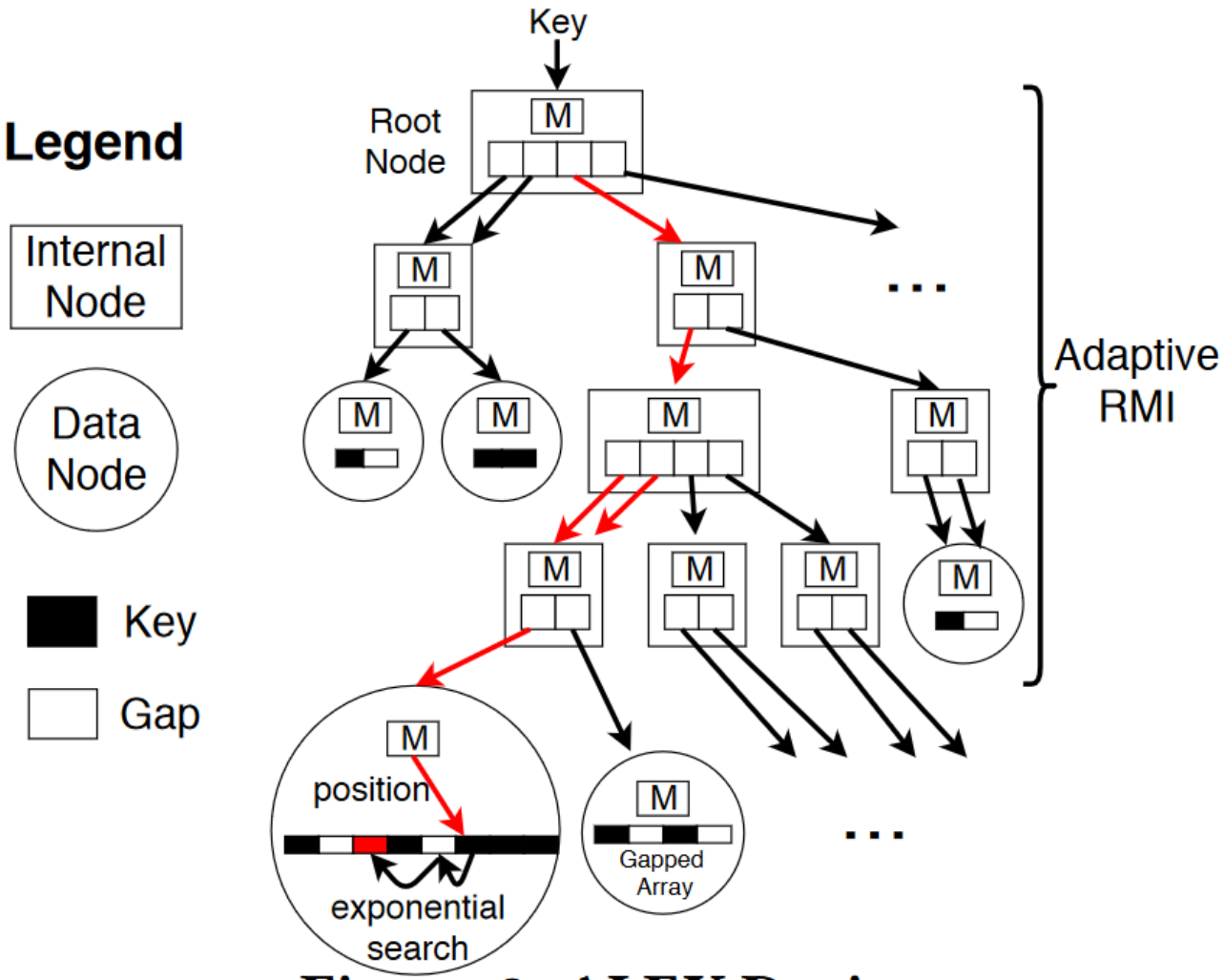


Figure 2: ALEX Design

17

[17] Изображение взято с архивной версии соответствующей статьи

ALEX: бенчмарк

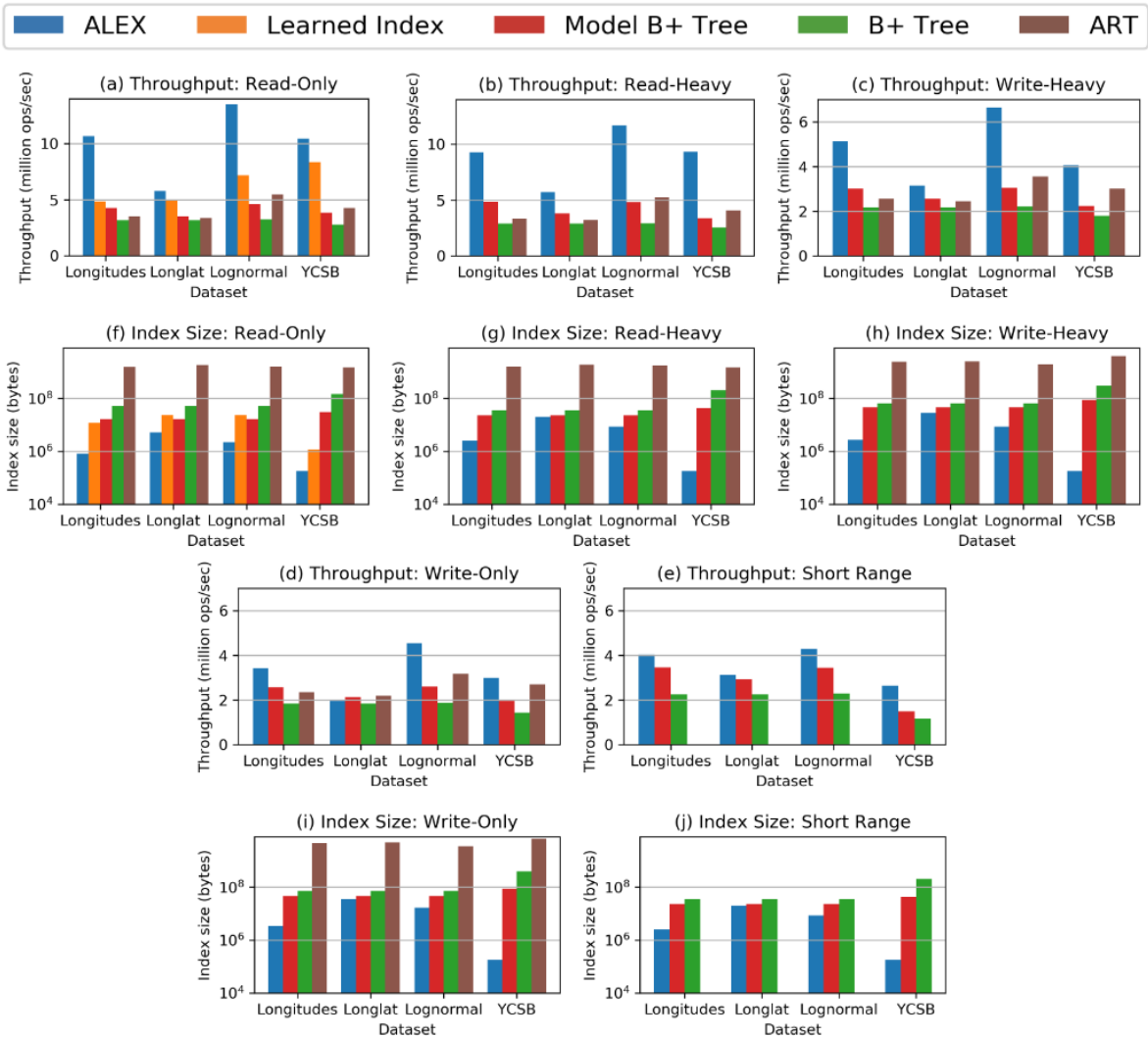


Figure 9: ALEX vs. Baselines: Throughput & Index Size. Throughput includes model retraining time.

18

[18] Изображение взято с архивной версии соответствующей статьи

ALEX: бенчмарк II

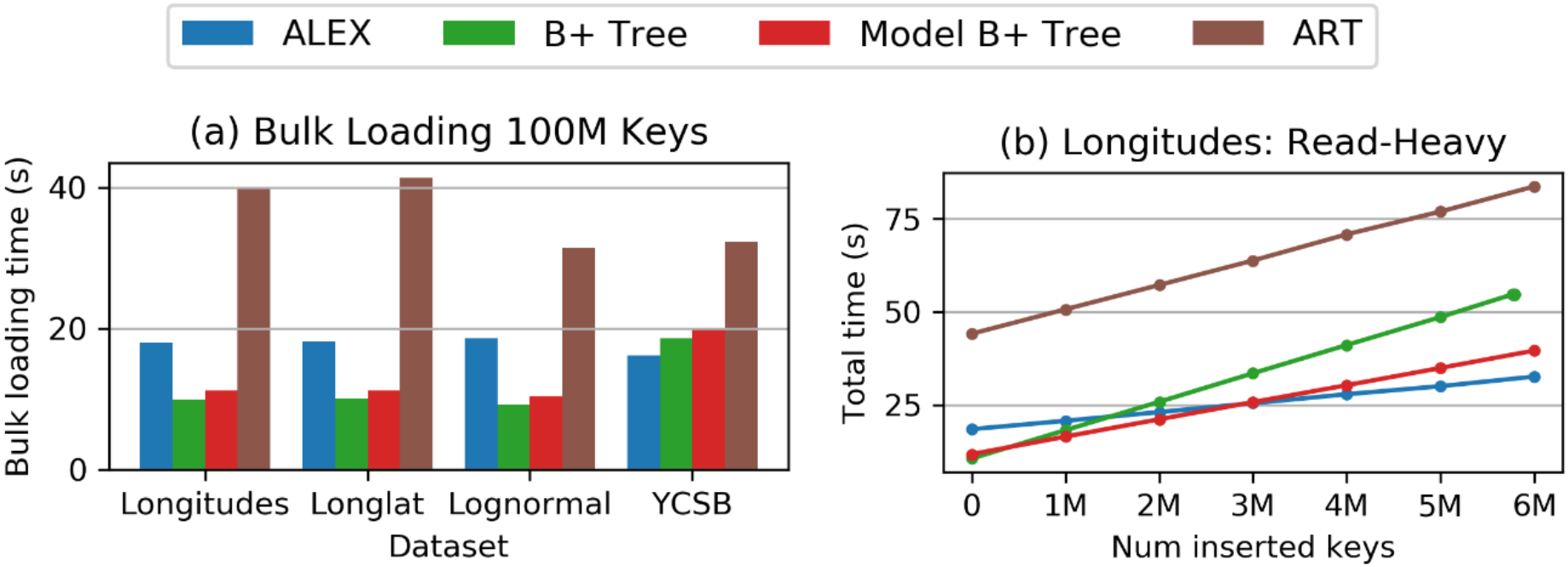


Figure 11: ALEX takes 50% more than time than B+Tree to bulk load on average, but quickly makes up for this by having higher throughput.

19

[19] Изображение взято с архивной версии соответствующей статьи

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.

Ссылки I

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.

Ссылки II


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 Návrat (Eds.). Springer-Verlag, London, UK, UK, 149–162.

Ссылки III

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.