Multidimensional indexing

Курс «Базы данных»

Антон Волохов

Yandex

2 декабря 2013 г.

Содержание

Содержание

Терминология

- **Точка** - Ряд базы данных, логический документ
- **Координата** - Значение в колноке базы данных
- **Расстояние** - $f(p_1, p_2) \rightarrow [0..1]$
- **DataSet** -Множество исходных точек
- **Домен** - Условие, пораждающее исходные данные

обозначения

- **DataSet** - P
- **Domain** - D
- **QueryPoint** - q
- **Distance measure** - $d(x_1,x_2)$
- Остальные обозначения появятся по мере продвижения

Disclaimer

- Если есть вопрос, меня можно и нужно перебивать
- Не на каждый вопрос я смогу ответить по ходу лекции
- Не на каждый вопрос я смогу ответить

Индекс

- **Индекс**- Избыточная структура данных, построенная над пространством P, ускоряющая поиск элементов из P.
- **Индексирование**- Процесс препроцессинга пространства, результатом которого является индекс.

Например

В Java, HashMap и TreeMap - простейшие индексы.

Обоснованность построения

Формально

- $\exists Q' \subset Q : \forall q \in Q' \tau(I, q) < \tau(P, q)$

Оценка качества

- Избыточность
- Сложность построения
- Деградация
- Сложность поиска
- Поддерживаемые классы запросов

Примеры запросов

Точное совпадение

Получить данные о сотруднике по его паспортным данным

Диапазон значений

Получить данные о всех сотрудниках в возрасте от 50 до 70 лет

Объединение значений

Получить данные о сотрудниках

Внешняя память

- Дорогой I/O
- Блочное чтение
- Потенциально бесконечные объемы данных

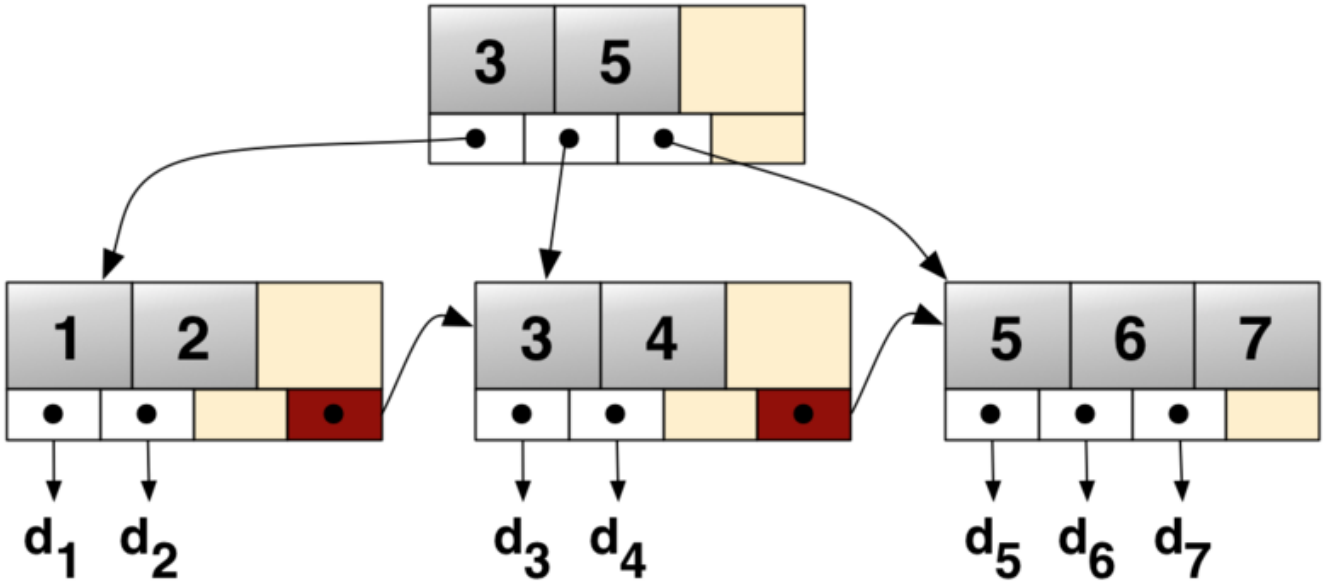
Особенности

- высокая ветвистость дерева
- строго заданные размеры узла

NB!

Оптимизация индексных структур для блочного чтения.

B+ Tree



B+ Tree

- Сбалансированное
- Данные только в листьях
- Ветвистое

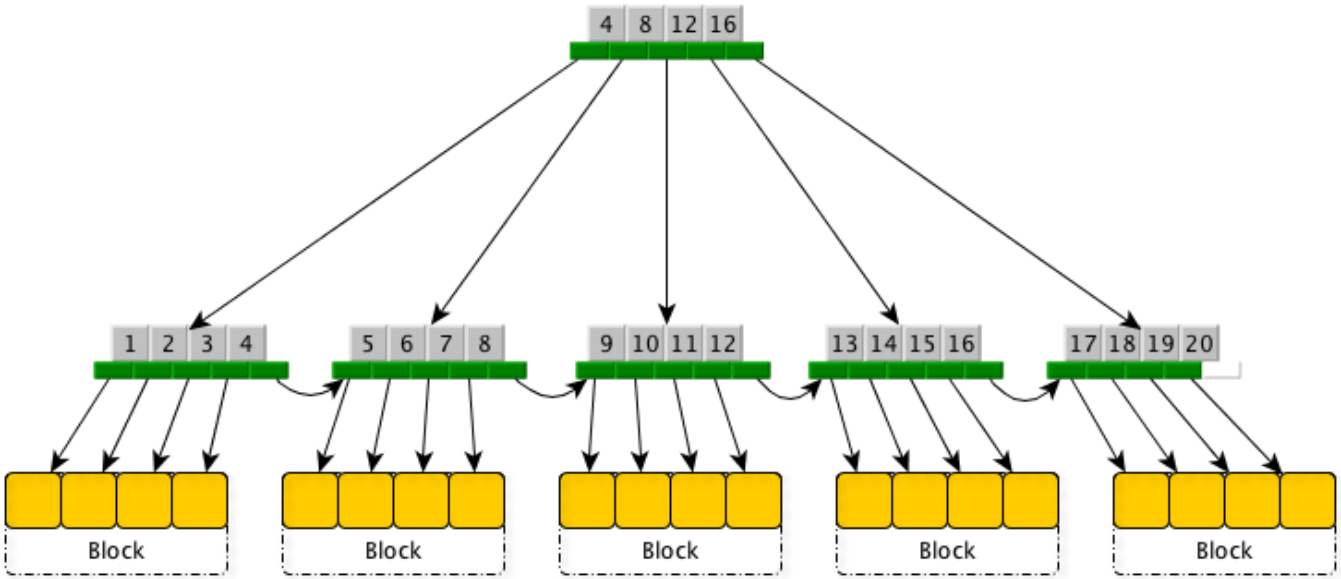
Преимущества

- Гарантированная производительность
- Малая высота дерева

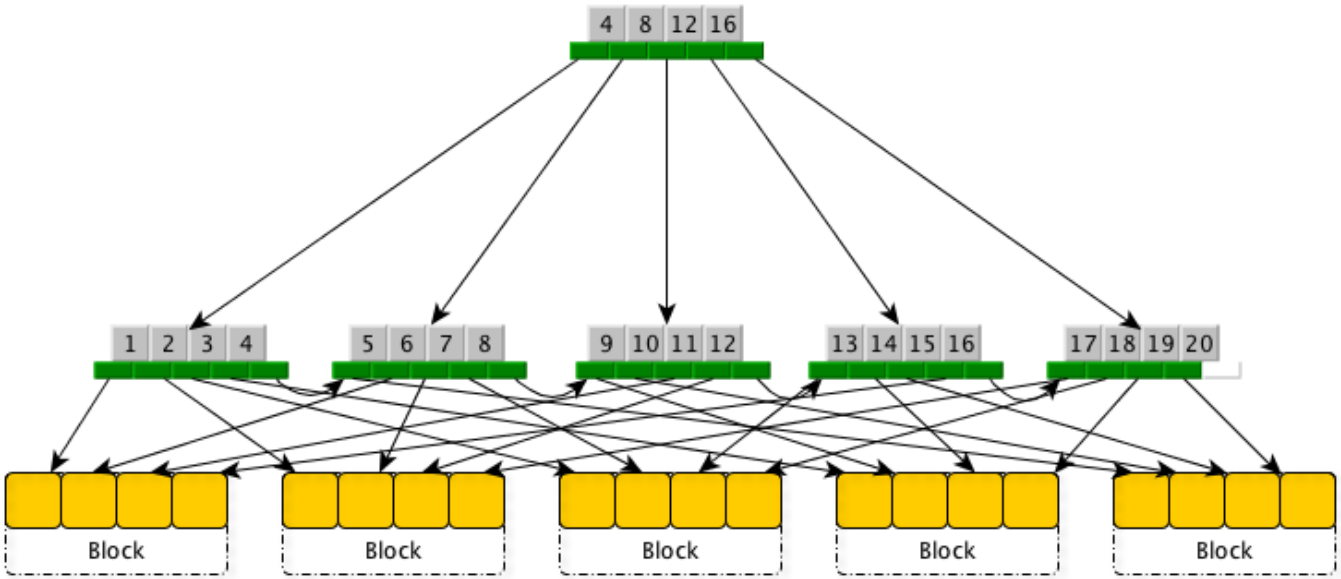
Недостатки

- Деградация в динамике

Деградация


4 8 12 16
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20
Block
Block
Block
Block
Block

Деградация


4 8 12 16
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
17 18 19 20
Block
Block
Block
Block
Block

Что можно искать?

Запрос на диапазон

$\forall q \in D, P$ retrieve $P' \subset P : \forall p' \in P', d(p', q) < \epsilon$

Запрос на поиск ближайших соседей

$\forall q \in D, P$ retrieve
$P' \subset P : \forall p' \in P', p'' \in P/P', d(p', q) < d(p'', q)$

Запрос на поиск похожих пар

P, retrieve $(p, p') \in P : d(p, p') < \epsilon$

Корректны для любого пространства, на котором задана функция расстояния.

Где можно искать

- Числа
- Строки
- Цепочки ДНК
- Тексты
- Изображения
- Видео
- etc.

Пример

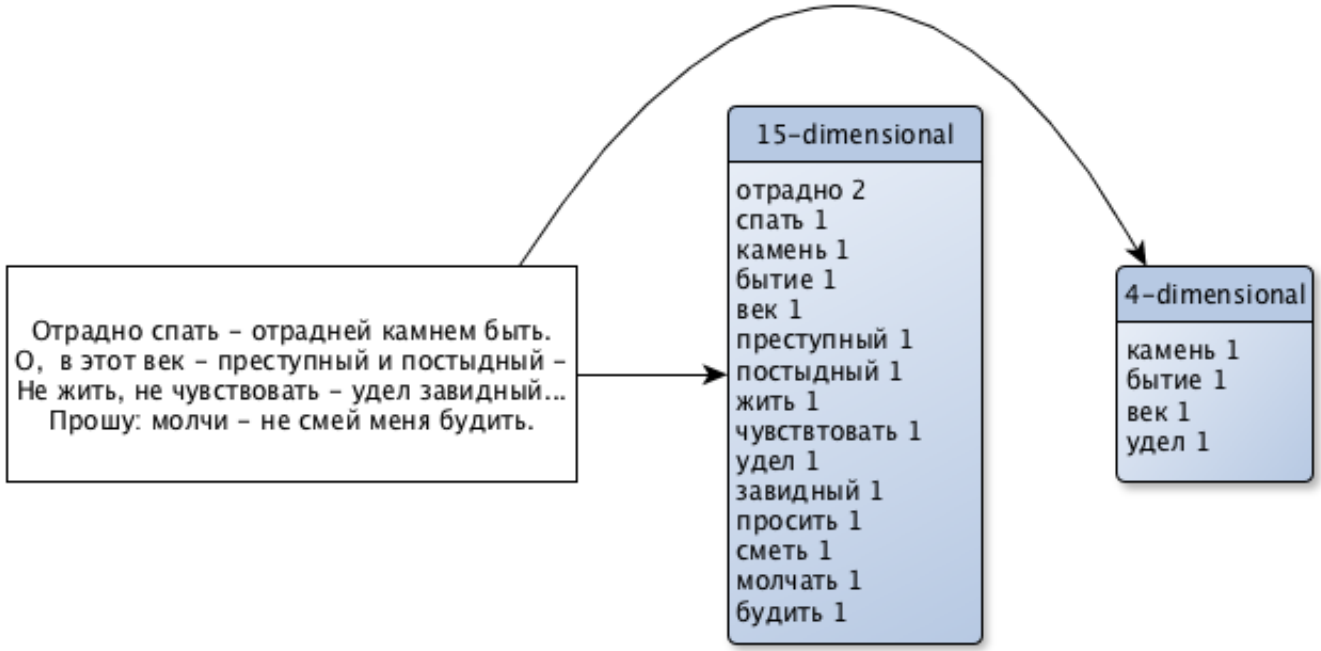
Задача - для некоторого текста q найти в базе все похожие на него.

Что необходимо знать для решения задачи

1. Представление текста в базе
2. Мера схожести текстов
3. Опционально - дополнительные сведения о запросе.

Представления Текстов

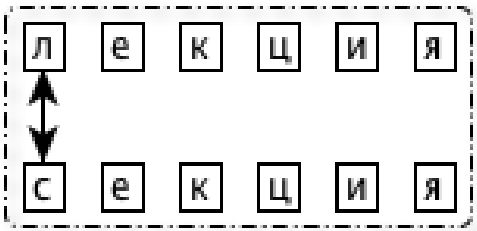
1. Облако тэгов.
2. Vector Text Model

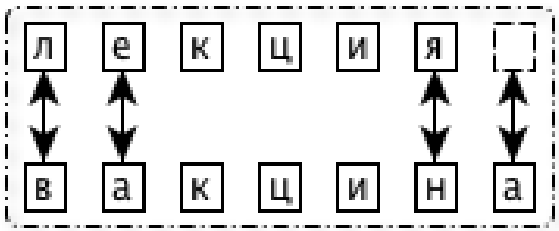


Меры схожести

- n-gram, $1 - (\cap ngram / \cup ngram)$

- Мера Левинштейна





- Мера Минковского

$$d_p(x, y) = \sqrt[p]{\sum |x_i - y_i|^p} \quad (1)$$

Абстрактное дерево поиска

1. добавление элемента
2. расщепление вершины
3. удаление элемента

Обобщаем B+ Tree

- Узел - прямоугольный параллелепипед с гранями вдоль гиперплоскостей базиса - **R-Tree**
- Ключ - центр сферы, объемлющей своих детей - **M-Tree**
- Ключ - гиперплоскость - **k-d Tree**
- etc

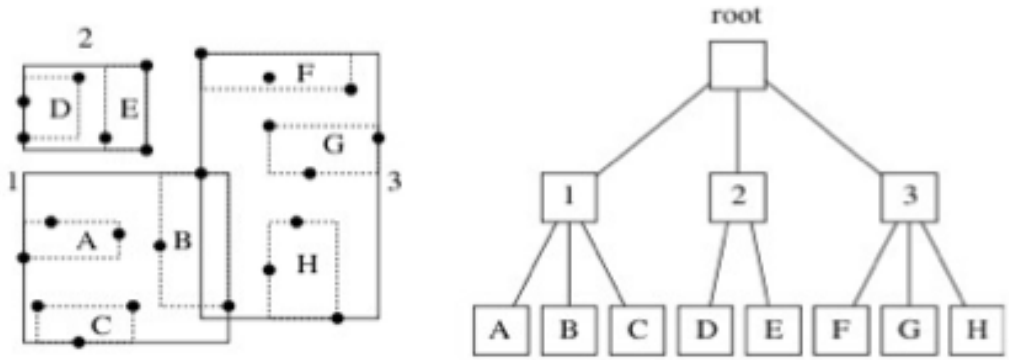
R-Tree

1. На каждом уровне спускаемся в ближайшую вершину, добавляем элемент. В случае необходимости, расщепляем вершину.
2. $split(Node) \to (Node_1, Node_2)$: $V(Node_1) + V(Node_2) \to min$; $remove(Node)$; $insert(Node_1)$; $insert(Node_2)$
3. На каждом уровне спускаемся в ближайшую вершину, удаляем элемент, если он есть. В случае необходимости удаляем вершину.

NB!

Узлы самопересекаются!

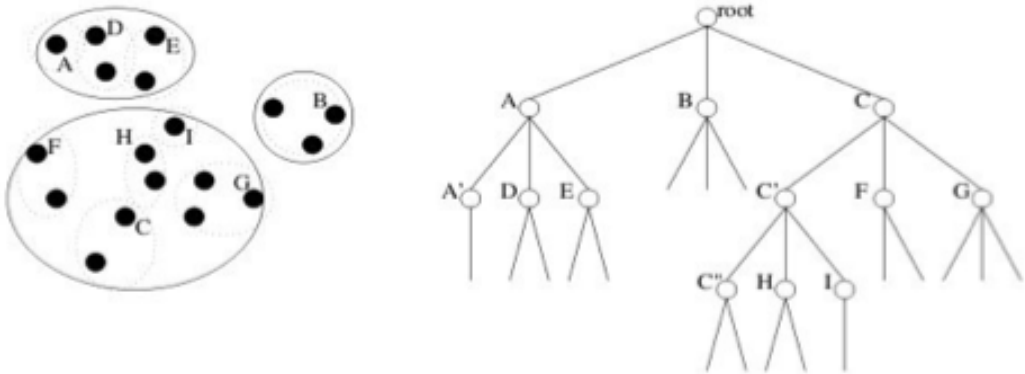
Оценка



- поиск: $O(log_m(n))$
- добавление: $O(log_m(n))$
- удаление: $O(log_m(n))$
- расщепление: $m - m^2$ в зависимости от алгоритма

m-tree

Примерно та же история

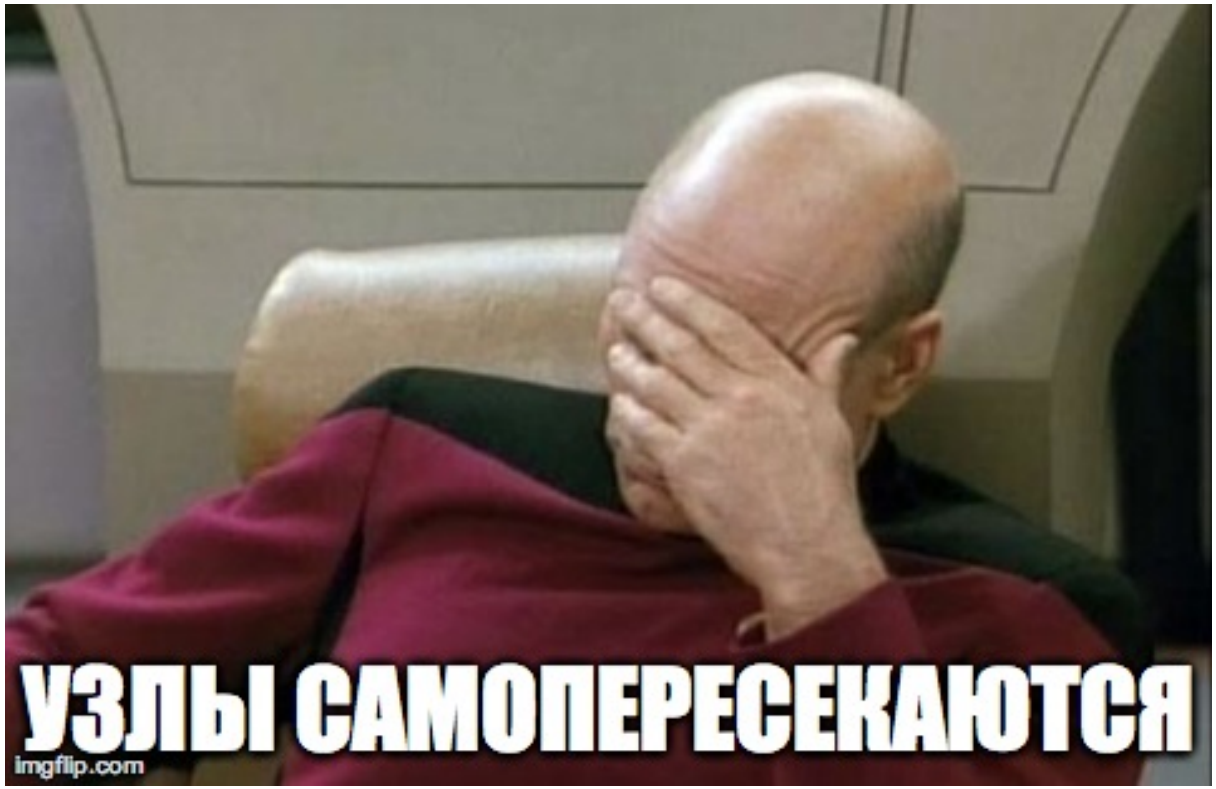


Кроме того, алгоритмы усложняются возможно тяжелым вычислением расстояния между точками.

Почему всё это не работает?

Проклятье размерности

Оценивая алгоритмы мы обошли одну важную вещь



Определение

- Пространство становится разреженным:

$$\lim_{d \to \infty} V(sphere)/V(parallelotope) \to 0 \qquad (2)$$

- Увеличивается вероятность самопересечения
- Теряется смысл мера Минковского.

Пример

- Дан единичный d-мерный куб, в котором равномерно распределены точки. Необходимо покрыть другим кубом 10
- $l^d = v$
- $l = v^{1/d}$

d=1

- $v = 0.1 \Rightarrow l = 0.1$

d=10

- $v = 0.1 \Rightarrow l = 0.8$

Деревья

- Каждый спуск совершается почти во все узлы
- В худшем случае медленнее, чем простой перебор.

Как починить деревья?

1. Настройка алгоритма расщепления узла
2. Выделение пространства меньшей размерности
3. Техники понижения размерности
4. Использование не деградирующих функций расстояния

Расщепление узла

- Использование кластеризации
- Переразбиение всего уровня

Поиск релевантного подпространства

Найти все классические произведения, похожие на роман Е. Замятина "Мы"

- Отсекаем неологизмы
- Отсекаем арго
- Возможно, уменьшаем выборку по годам или авторам

Понижение размерности

- Построение облака тэгов
- Поиск репрезентативных точек (кластеризация)

Вероятностное понижение размерности

Для пространства P размерности n, построить такое биективное отображение в пространство P' размерности k, $k << n$, что если две точки близки в P', то вероятность того, что они близки в P больше ϵ

Проекция двумерного пространства на одну из осей - простейшее понижение размерности.

Хеш

Хешевый индекс

- Поддерживает запросы только на совпадение
- Равномерно распределяет точки по корзинам

Как адаптировать хеш для наших многомерных нужд?

Поменять функцию так, что она складывала близкие точки не в разные, а в одну корзину.

Locality-Sensitive Hashing

Определение

Семейство хеш функций называется (R, cR, P_1, P_2)-чувствительным, если: $\forall p, q \in$ domain D, threshold R $d(p, q) \leq R \Rightarrow Pr(h(p) = h(q)) \geq P_1$ $d(p, q) \geq cR \Rightarrow Pr(h(p) = h(q)) \leq P_2$

- Определение имеет смысл при $P_1 > P_2$
- Вероятностные гарантии коллизии близких точек

Индексирование

Домен D - унарные вектора, длины не более M, R=1, c=2

1.
   - Хеш функция - наличие единицы на i-м месте
   - $P_1 = 1/2$; $P_2 = 1/4$; $P_{err} = 1/2$
2. Теперь попытаемся увеличить разность P_1 и P_2
3.
   - Хешируем нашу точку с помощью k случайных хеш функций.
   - $P_{1k} = (1/2)^k$; $p_{2k} = (1/4)^k$; $P_{err} = 1 - P_{1k}$
4. Теперь увеличиваем вероятность коллизии.
5.
   - Создаём n таких композитных индексов
   - $P^n_{1k} = n * P_{1k}$; $P^n_{2k} = n * P_{2k}$; $P_{err} = 1 - P^n_{1k}$
   - Изменяя k и n, можно минимизировать P_{err}

Литература

- P. Indyk, R. Motwani. “Approximate nearest neighbor: towards removing the curse of dimensionality”. Proceedings of the Symposium on Theory of Computing, 1998.
- Guttman: R-Trees: “A Dynamic Index Structure for Spatial Searching”. SIGMOD Conference 1984: 47-57, 1984.

Литература

- Skopal, T. et al., “Revisiting M-Tree building principles”. Proceedings of the 7th East European Conference on Advances in Databases and Information Systems (ADBIS), 2003.
- Similartiy Search: The Metric Space Approach Series: Advances in Database Systems, Vol. 32 Zezula, P., Amato, G., Dohnal, V., Batko, M. 2006, XVII

Вопросы?

- Почта a.v.volokhov@gmail.com