Курс «Базы данных»

Тема: Физическое проектирование БД. Индексы

Барабанщиков
Игорь Витальевич

План лекции

1. Алгоритмы поиска данных, используемые СУБД.
2. B+-деревья.
3. Индексы СУБД Oracle.
4. Рекомендации по использованию индексов.

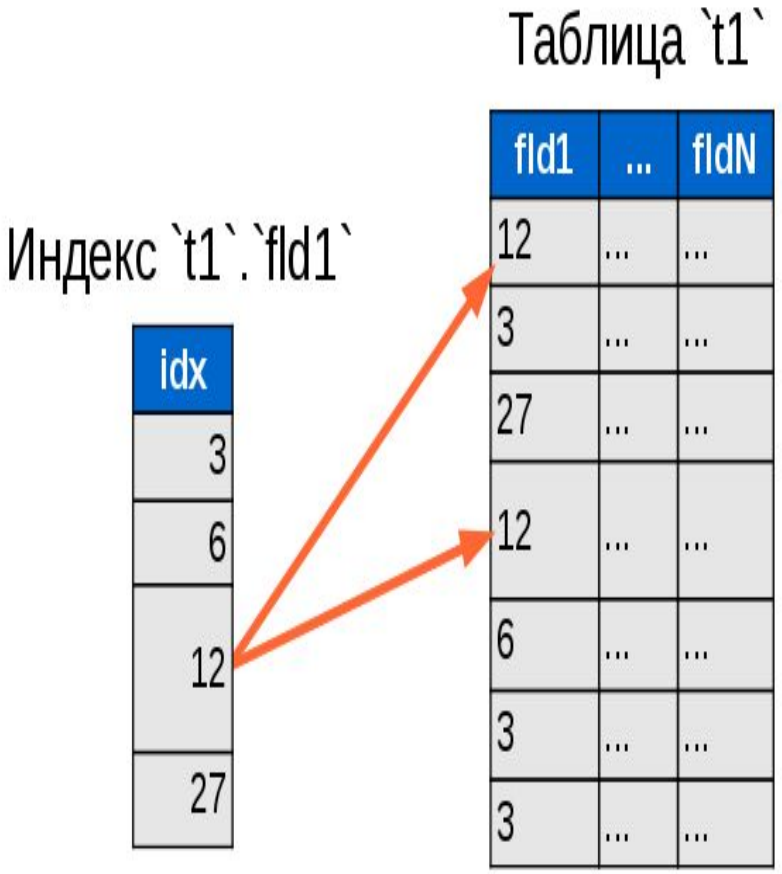
Поиск данных в БД

- В БД часто используются операции **поиска** данных.

  Пример: Найти всех студентов с фамилией Кио.
- **Просмотр всех записей в таблице использовать неэффективно** – долго работает.
- **Для ускорения поиска** в базах данных используют вспомогательные объекты: **индексы**.

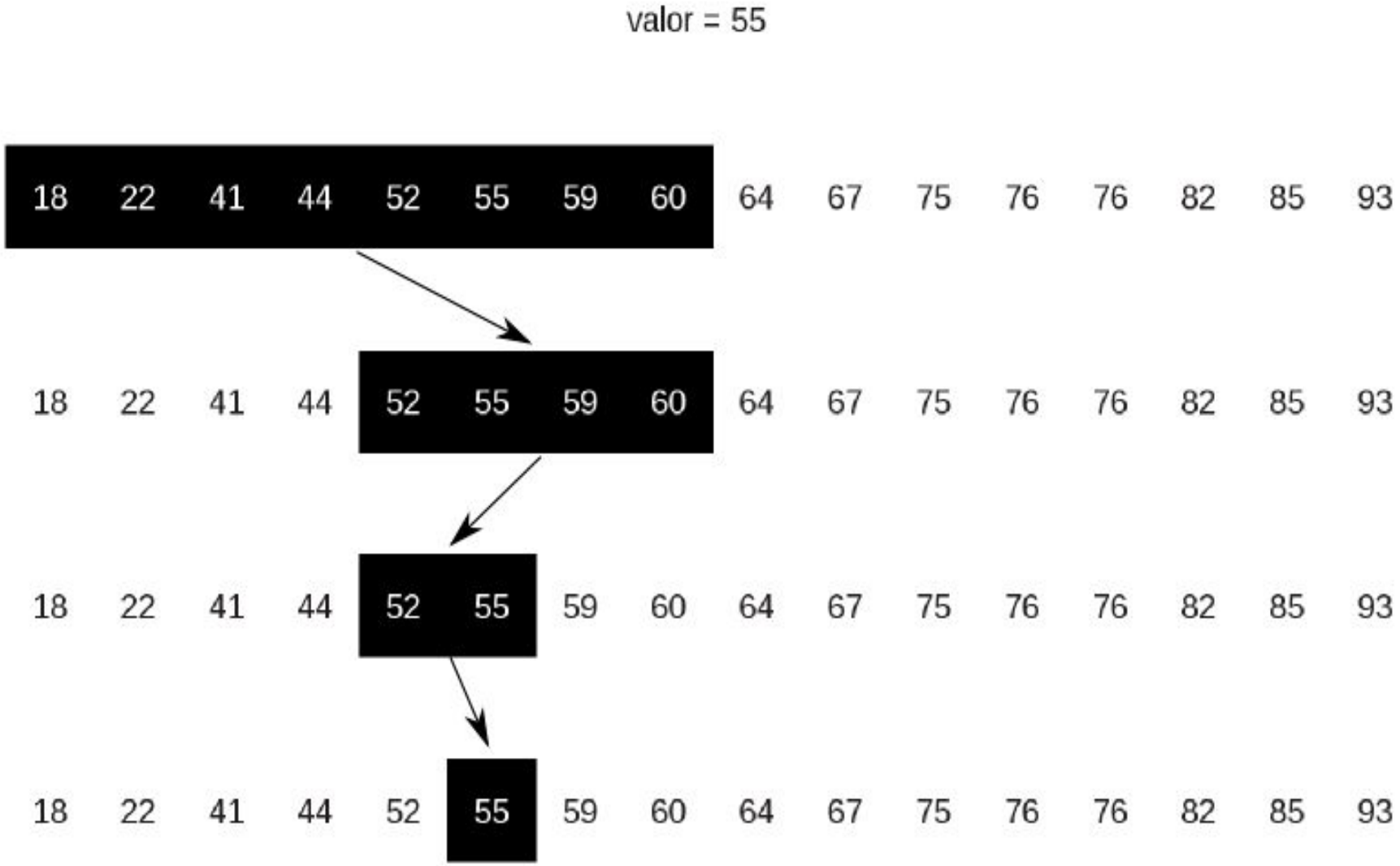
Индекс

Индекс – это структура, которая **определяет соответствие значения ключа записи** (атрибута или группы атрибутов) **и местоположения этой записи** в таблице.



В основе работы индекса лежит

Бинарный поиск

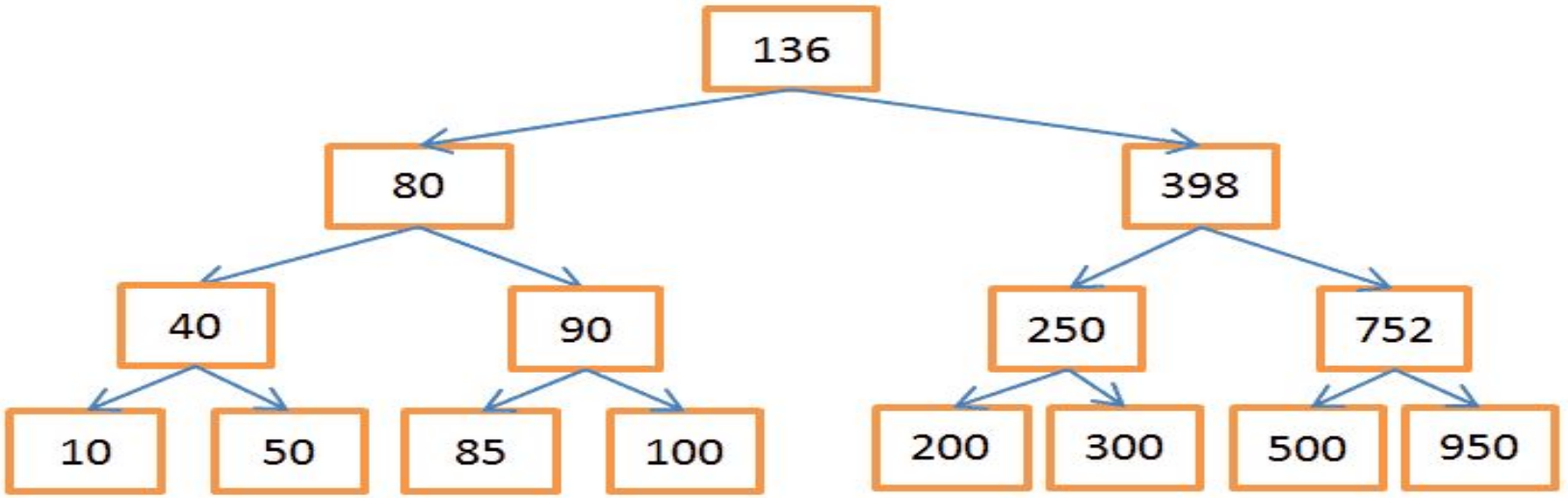


Бинарное дерево поиска

Бинарное дерево поиска — это дерево, в котором ключ в каждом узле должен быть:

- Больше, чем любой из ключей в ветке слева.
- Меньше, чем любой из ключей в ветке справа.

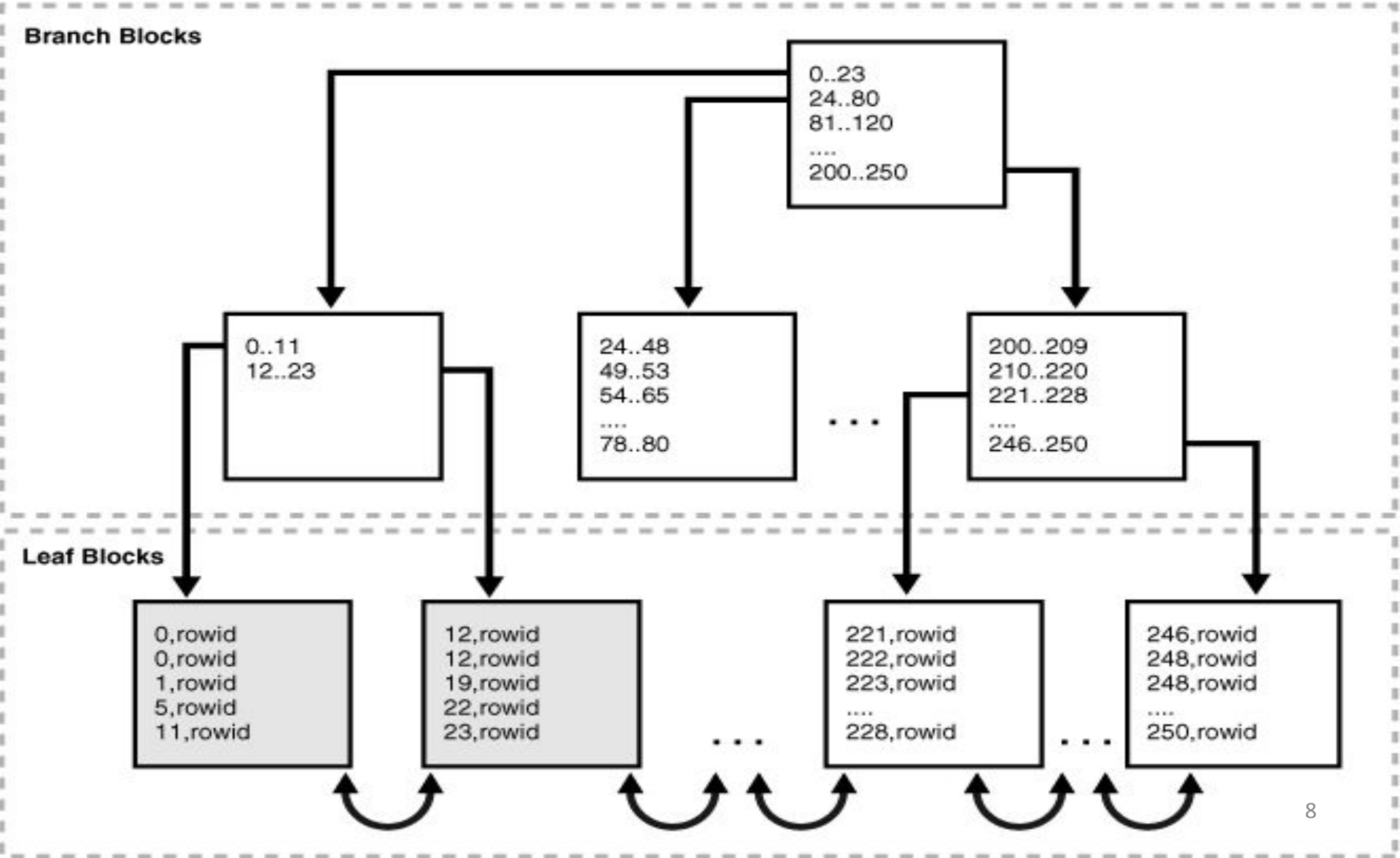
Несмотря на высокую скорость поиска, дерево плохо работает в случаях, когда нужно получить несколько элементов в пределах двух значений (BETWEEN a AND b).



B+дерево

- Нужен *более эффективный* способ **запроса диапазона**.
- В современных БД для этого используется *модифицированная* версия дерева — **B+дерево.**
- Его особенность в том, что **информацию (ID строк связанной таблицы) хранят лишь самые нижние узлы («листья»),** а все остальные узлы предназначены для оптимизации поиска по дереву

B+tree индекс Oracle



Сканирование B+tree индекса

- **Листовые блоки содержат по 2 элемента:**

  - *индексированные значения столбца*

  - *идентификатор ROWID строки*, которая содержит это значение столбца.
- **ROWID** – уникальный указатель Oracle, определяющий физическое местоположение строки и обеспечивающий самый быстрый способ доступа к строке в БД Oracle.
- **Сканирование индекса быстро дает ROWID строки**, и отсюда можно быстро получить доступ к ней.
- **Большинство B-деревьев имеет всего три и менее уровней**.
- **Для выполнения поиска по B-дереву**

Сбалансированное дерево

- В общем случае получим некоторое дерево, **каждый родительский блок которого связан с одинаковым количеством подчиненных блоков**, число которых равно числу индексных записей, размещаемых в одном блоке.
- **Количество обращений к диску при этом для поиска любой записи одинаково и равно количеству уровней в построенном дереве**.
- Такие деревья называются **сбалансированными (balanced)** именно потому, что путь от корня до любого листа в этом древе одинаков.
- **Именно термин "сбалансированное" от английского "balanced" — "сбалансированный, взвешенный" и дал название данному методу**

Индексы СУБД Oracle

- **B*Tree** индексы:

  - наиболее часто используемый тип индекса
- **Reverse** – индексы
- **Индексы, основанные на функциях** (function-based)
- **Bitmap**-индексы
- **Составные** индексы

Создание индексов

- Индексы создаются в БД с помощью команды **CREATE INDEX**.
- Создание *обычного* индекса

```
CREATE INDEX idx_emp
    ON emp(last_name);
```

- Создание *уникального* индекса

```
CREATE UNIQUE INDEX idx_id
    ON emp(emp_id);
```

Reverse – индексы

- **Reverse index** – это тоже B-tree индекс, но с обратным порядком байтов.
- Благодаря перестановке байтов **две соседние записи индекса попадают в *разные* блоки индекса.**
- Используются в основном для монотонно возрастающих значений **с целью снятия конкуренции за последний листовой блок индекса.**
- **Не может использоваться для диапазонного поиска.**
- Когда в запросе присутствует предикат неравенства, ответ получается медленнее.

Пример Reverse-индекс

Поле в таблице (bin)	Ключ reverse-индекса (bin)
0000 0001	0001 0000
…	…
0000 1001	1001 0000
0000 1010	1010 0000
0000 1011	1011 0000

Значение в индексе изменяется намного больше, чем само значение в таблице, и поэтому **в структуре**

B-Tree, они попадут в разные блоки.

Пример

- Программа продажи билетов на поезда
- В таблицу TICKET (билет) каждую секунду вставляется много новых записей.
- **При наличии обычного индекса возникает конкуренция** за самый правый листовой блок индекса.
- Для решения проблемы надо создать реверсивный индекс:

```
CREATE  INDEX  ticket_idx
      ON  ticket(ticket_id)   REVERSE;
```

Function-based Index

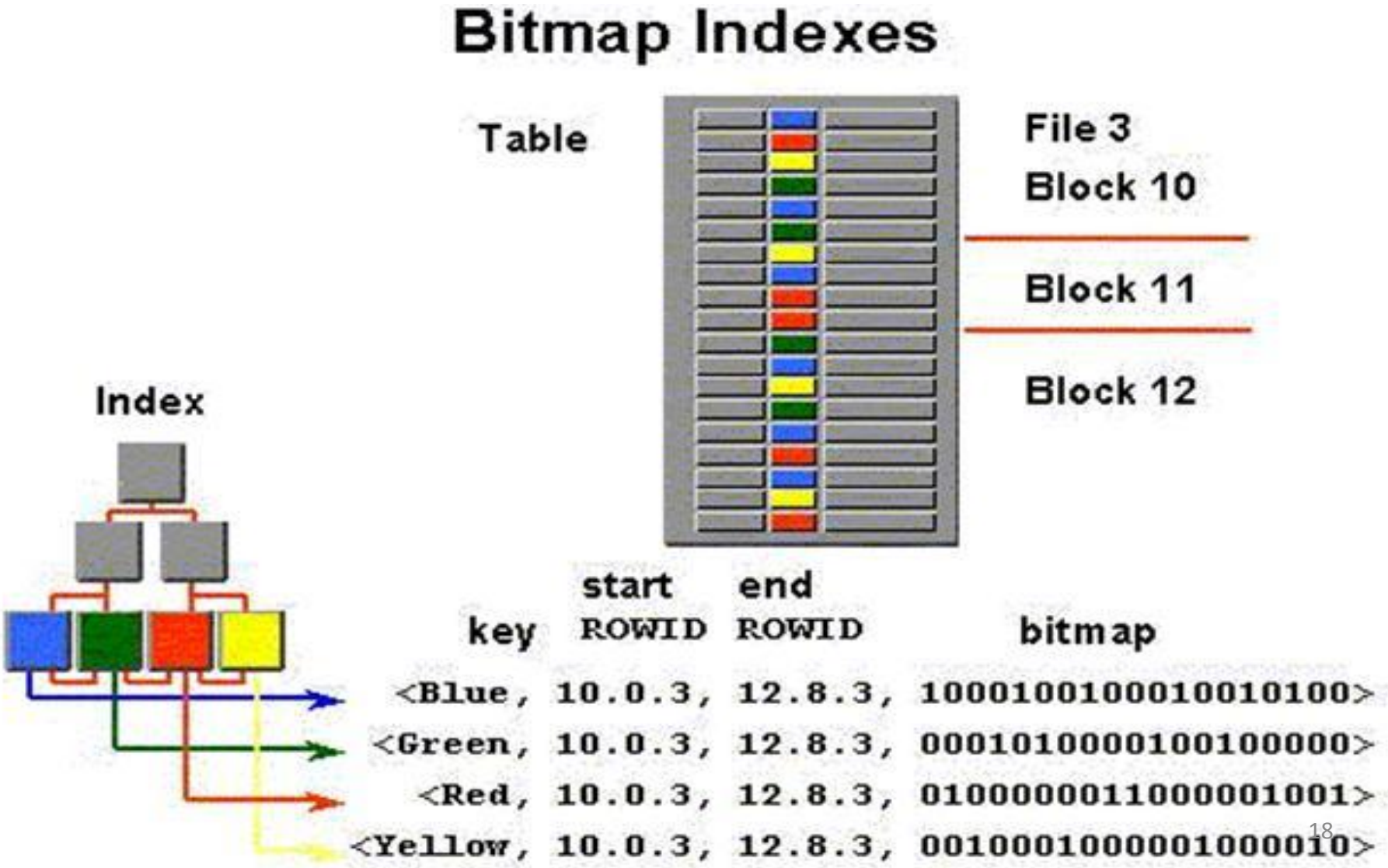
- **Индексы на основе функций предварительно вычисляют значения функций по заданному столбцу и сохраняют результат в индексе.**
- Функциональные индексы часто строятся для полей, значения которых проходят предварительную обработку перед сравнением в команде SQL.

```
-- создаем индекс на основе функции UPPER
CREATE INDEX firstname_idx ON emp(UPPER(fname));
-- выполняем запрос
SELECT * FROM emp
   WHERE UPPER(fname) = ‘ИВАН’;
```

Bitmap Index

- **Bitmap index** – содержит отдельные битовые карты для каждого возможного значения столбца.
- Каждому биту соответствует строка с индексируемым значением.
- Если значение бита равно 1 – значит запись, соответствующая позиции бита, содержит индексируемое значение для данного столбца.
- Применяются, в основном, в **OLAP-системах**.
- Это идеальный индекс для столбца с **низкой селективностью** (число уникальных записей в таблице мало) при большом размере таблицы.
- Чувствительны *к перестроению индекса*.

Структура bitmap-индекса



Пример BITMAP-индекса

В таблице EMP есть поле SEX (пол), которое обладает **низкой селективностью** – может принимать всего два значения.

```
CREATE BITMAP INDEX idx_sex ON emp(sex);
```

Битовый индекс – не слишком разумная альтернатива для таблиц, подвергающихся большому количеству вставок, удалений и обновлений.

Советы по работе с индексами

Создавайте индексы на следующие поля:

- **первичный ключ**, такой индекс создается автоматически;
- **внешний ключ** или поле, которое часто используется для связи таблиц.
- поле, используемые для частого поиска ряда значений (WHERE);
- поле, по которому сортируются данные (ORDER BY, UNION, DISTINCT);
- поля, которые группируются во время агрегации (GROUP BY);
- поле, которое часто используется в запросах SELECT.

Индекс имеет смысл, если нужно обеспечить доступ одновременно не более чем к 4-5% данных таблицы.

Советы по работе с индексами

Не стоит создавать индексы на поля если:

- **Столбцы редко используются в запросе**;
- **Столбцы содержат значения с низкой селективностью (неуникальные**). Исключения для низкой селективности составляют случаи, при которых выборка чаше производится по редко встречающимся значениям
- **Столбцы состоят из длинно-символьных строк или BLOB**. Колонки с этими типами данных не могут быть проиндексированы.
- **Столбцы часто обновляются**, т.к. команды обновления ведут к потере времени на обновление индекса
- **Таблица сравнительно небольшая**. Для таких таблиц больше подходит полное сканирование. В случае маленьких таблиц нет необходимости в хранении данных и таблиц и индексов

Составные индексы

Составной индекс включает 2 и более столбца одной таблицы.

В некоторых случаях использование составного индекса предпочтительнее, чем одиночного:

- Если в запросах часто используются только столбцы, участвующие в индексе, система может вообще не обращаться к таблице для поиска данных.
- **Несколько столбцов с низкой селективностью в комбинации друг с другом могут дать гораздо более**

Применение составных индексов

CREATE TABLE emp(id, name, sex, hobby, age)

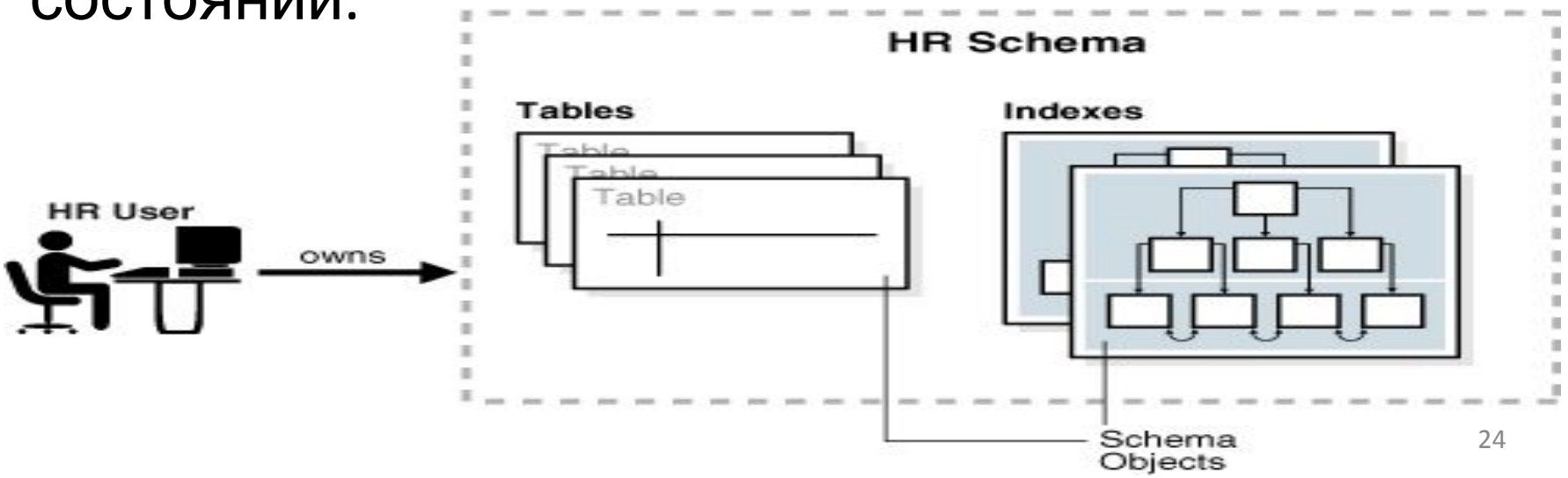
CREATE INDEX i_emp ON emp(hobby, age, sex)

Обращение к составному индексу возможно только, **если в условиях выбора участвуют столбцы, представляющие собой лидирующую часть составного индекса**.

Обращение к индексу будет происходить в тех случаях, когда в условии запроса участвуют поля (hobby, age, sex), (hobby, age)

Использование индексов

- Каждый индекс **связан с определенной таблицей**
- Индекс обычно **хранится отдельно** от таблицы
- **СУБД использует индексы неявно** при выполнении команд SQL.
- **СУБД поддерживает индексы** в актуальном состоянии.



Итоги

- При разработке эффективных приложений важным этапом является *физическое проектирование* БД.
- Одной из задач физического проектирования БД является создание индексов.
- **СУБД Oracle предоставляет богатые возможности для выбора индексов.**