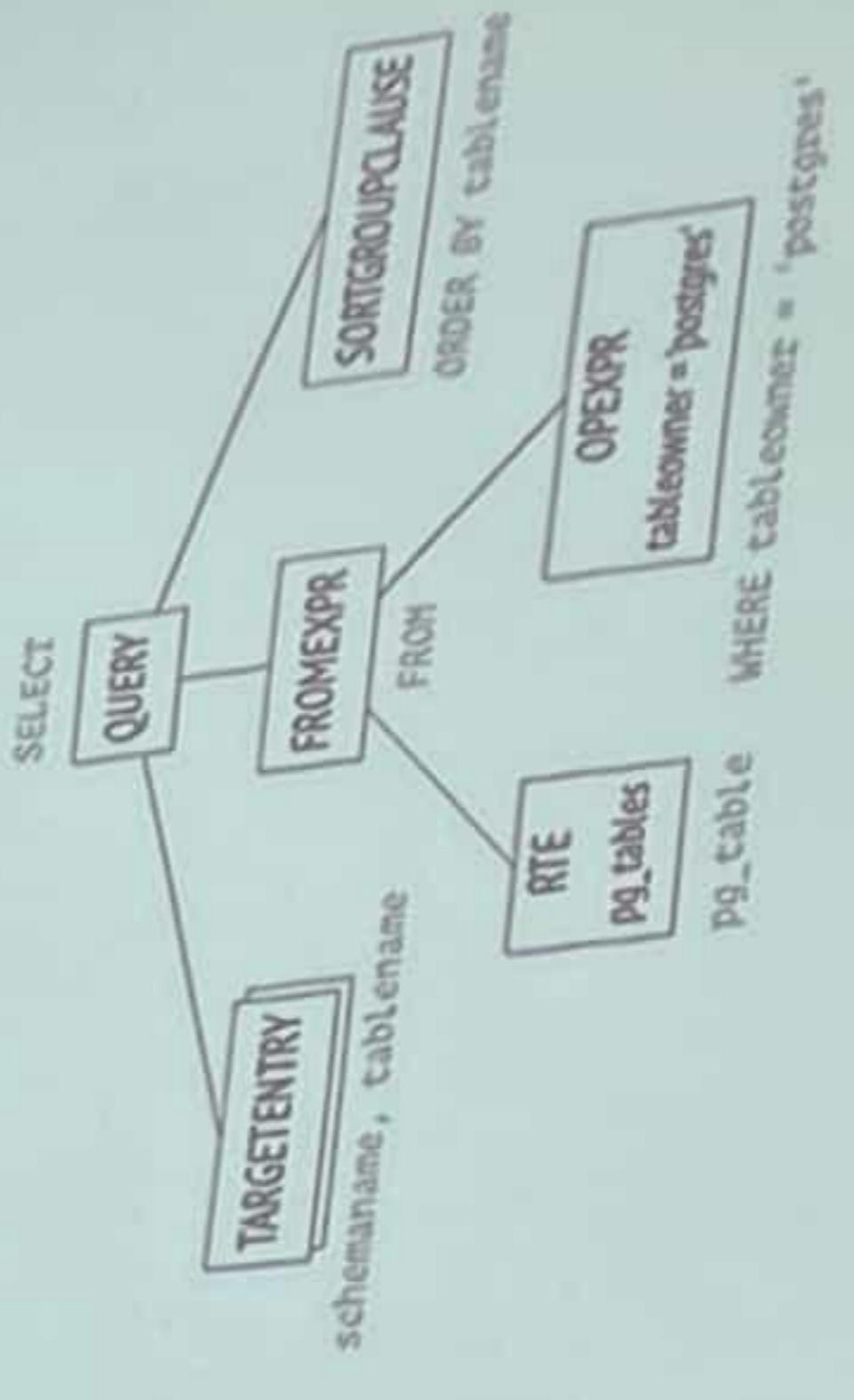


## Этапы выполнения запроса

### Лексический разбор

- На этом этапе сервер принимает SQL-текст запроса и на наличие синтаксических ошибок. Результатом является дерево разбора (parse tree), представляющее структуру запроса.



## Этапы выполнения разбора

### Семантический разбор

- Заменяет представления на их определения или обрабатывает правила, определённые для Таблиц.
- Проверка прав у пользователя к этим объектам.
- Вся необходимая для семантического анализа информация хранится в системном каталоге.
- Семантический анализатор получает от синтаксического анализатора дерево разбора и перестраивает его, дополняя ссылками на конкретные объекты базы данных, указанием типов данных и другой информацией.

## **Этапы выполнения планирования запроса**

- Любой запрос можно выполнить разными способами.
- План выполнения также представляется в виде дерева, но его узлы содержат не логические, а физические операции над данными.
- Текстовое представление плана выводит команда EXPLAIN

## Этапы выполнения запроса

### Планирование

- Планировщик (planner) принимает переписанное дерево запроса и генерирует один или несколько вариантов выполнения запроса.
- Оптимизатор оценивает стоимость каждого варианта плана, используя статистические данные о таблицах, и выбирает наиболее выгодный план.

## Этапы выполнения запроса

### Планы

- Количество возможных планов экспоненциально зависит от количества соединяемых таблиц.
- Для сокращения пространства перебора традиционно используется алгоритм динамического программирования в сочетании с некоторыми эвристиками.
- Точное решение задачи оптимизации не гарантирует, что найденный план действительно будет лучшим.

# Этапы выполнения запроса

## Общие и частные планы

```
=> EXECUTE plane('763');
=> EXECUTE plane('763');
=> EXECUTE plane('773');
=> EXPLAIN EXECUTE plane('319');

-----  
Seq Scan on aircrafts_data_ml  
Filter: ((aircraft_code)::text = '319'::text)  
(2 rows)

-----  
QUERY PLAN  
Seq Scan on aircrafts_data_ml  
Filter: ((aircraft_code)::text = '319'::text)  
(2 rows)

-----  
EXPLAIN EXECUTE plane('320');
=> EXPLAIN EXECUTE plane('321');

-----  
Seq Scan on aircrafts_data_ml  
Filter: ((aircraft_code)::text = '$1')  
(2 rows)
```

## Этапы выполнения запроса

- После выбора оптимального плана последовательность операций, предусмотренных планом, запускает буферный менеджер, который минимизирует количество операций ввода-вывода на диск.
- Если запрос рассчитан на параллельное выполнение, результат собирается из нескольких рабочих процессов и объединяется в итоговый набор данных.

## Статистика Определение

- это набор данных, который собирается системой для оценки распределения значений в таблицах и столбцах, что затем используется планировщиком запросов для выбора наиболее эффективного плана выполнения.

## Статистика хранение

- PostgreSQL pg\_statistic хранит собранную статистику в системной таблице
- Прямой доступ к pg\_statistic обычно не требуется, и для просмотра статистики чаще используют представление pg\_stats
- Это публичное представление, которое предоставляет информацию из pg\_statistic в удобном для чтения виде, без избыточных деталей.
- Базовая статистика уровня отношения хранится в таблице pg\_class

# Статистика

## Данные

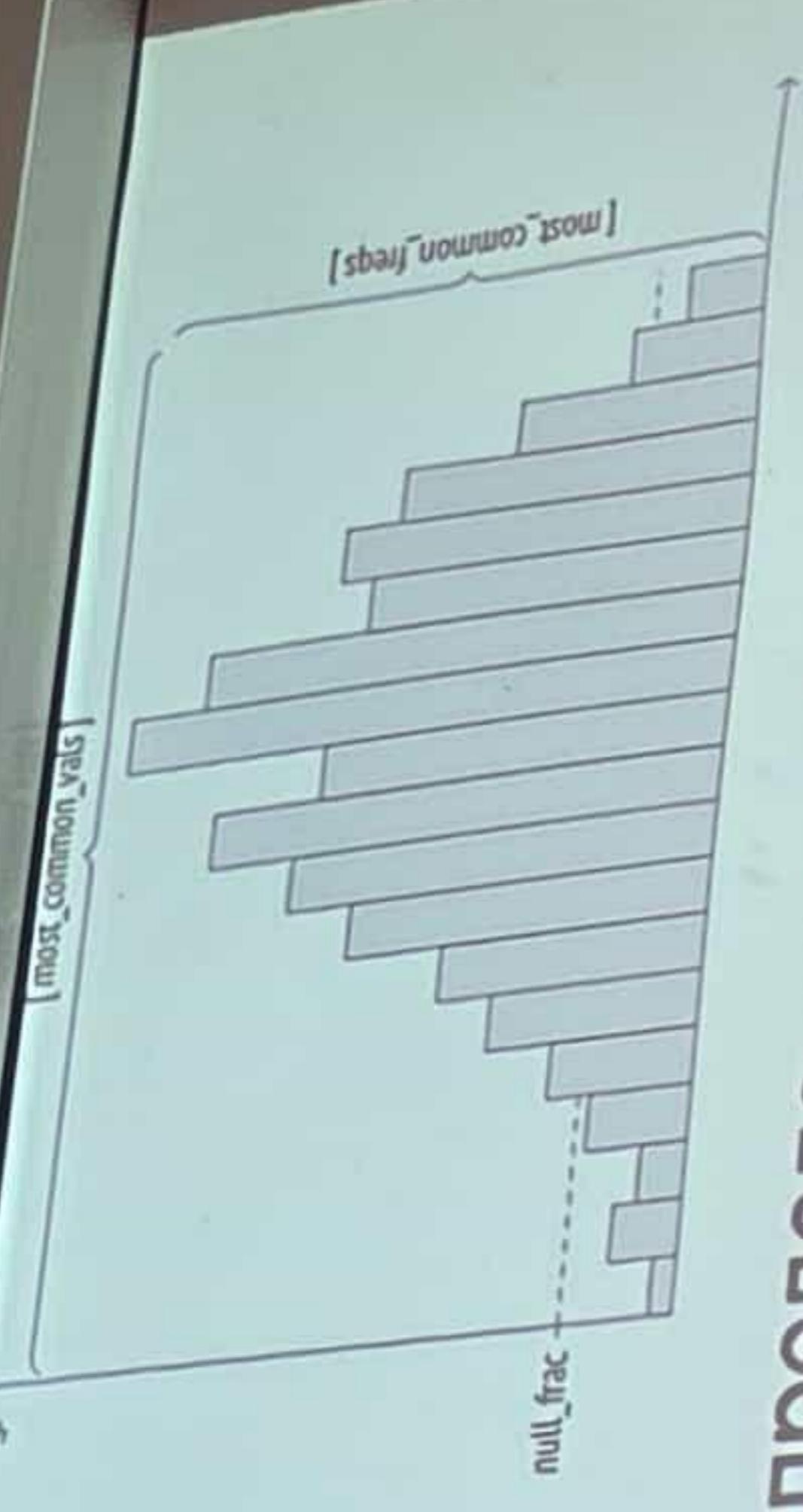
- `reltuples`: число кортежей в отношении.
- `n_distinct`: оценка количества уникальных значений.
- `most_common_vals` и `most_common_freqs`: список наиболее часто встречающихся значений и их частоты.
- `histogram_bounds`: границы гистограммы распределения значений, которые используются для оценки селективности диапазонных запросов.
- `correlation`: оценка корреляции между порядком значений в столбце и порядком их хранения.
- `null_frac`: Доля неопределенных значений

## Статистика Сбор статистики

- Статистика собирается при анализе, ручном или автоматическом.  
Однако ввиду особой важности, также при выполнении некоторой базовой статистике CREATE INDEX и REINDEX и некоторых операций (VACUUM FULL и CLUSTER, ANALYZE) уточняется при очистке.
- Для анализа случайно выбираются 300 × default\_statistics\_target строк, заданной точности, слабо зависит от объема анализируемых данных, поскольку размер выборки, достаточный для построения статистики, размер таблицы не учитывается.

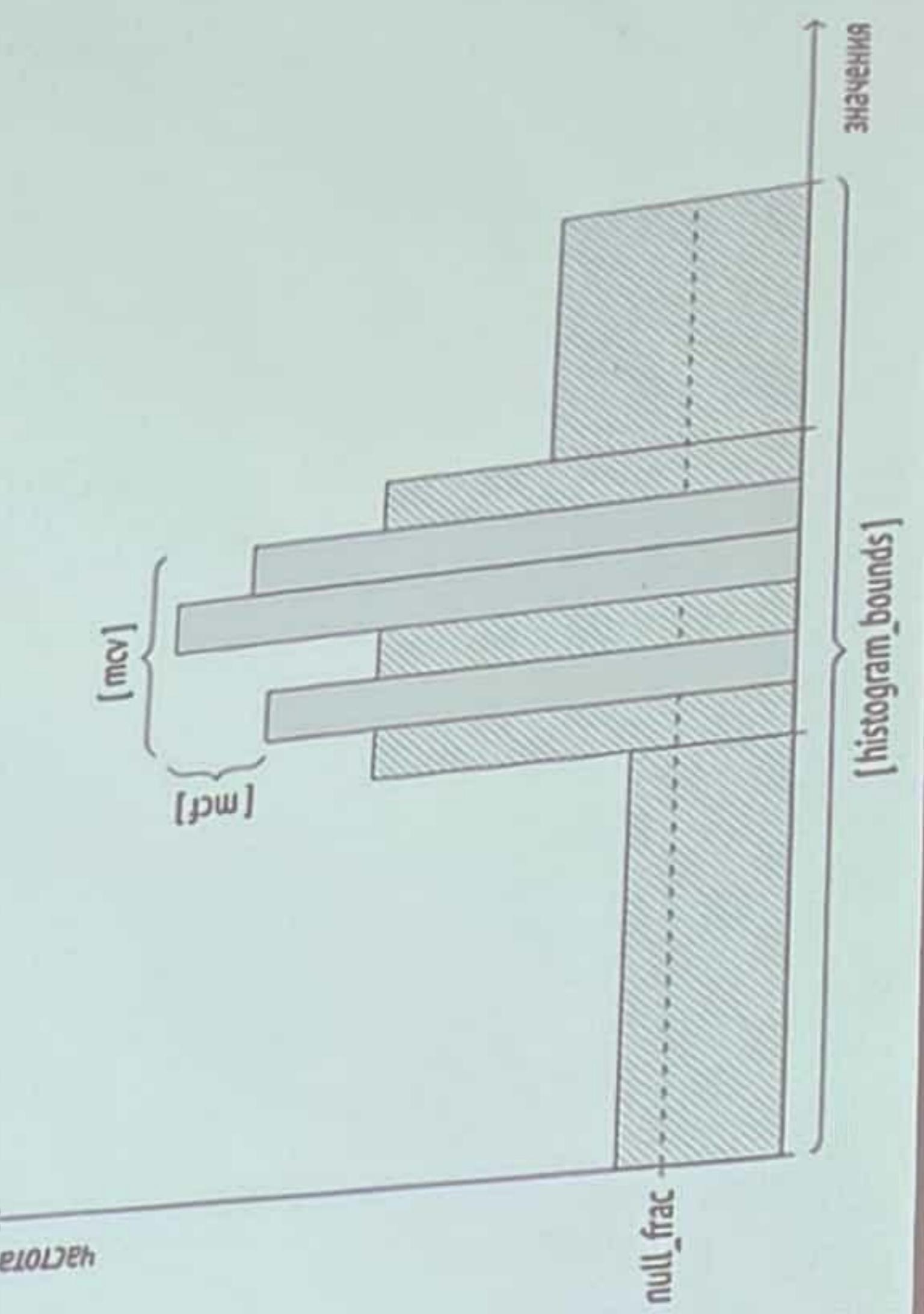
## Статистика pg\_stats

- Для уточнения оценки при неравномерном распределении собирается статистика по наиболее часто встречающимся значениям и частоте их появления. Представление pg\_stats показывает два этих массива в столбцах most\_common\_vals и most\_common\_freqs.
- Список частых значений используется и для оценки селективности условий с неравенствами. Например, для условия вида «столбец < значение» надо найти в most\_common\_vals все значения, меньшие искомого, и просуммировать частоты из most\_common\_freqs.



## Статистика pg\_stats

- Когда число различных значений слишком велико, чтобы записать их в нескольких корзин, в которых хранится диапазон. Гистограмма состоит из ограниченено всем же параметром default\_statistics\_target.
- Ширина корзин выбирается так, чтобы в каждую попало примерно одинаковое количество количества значений

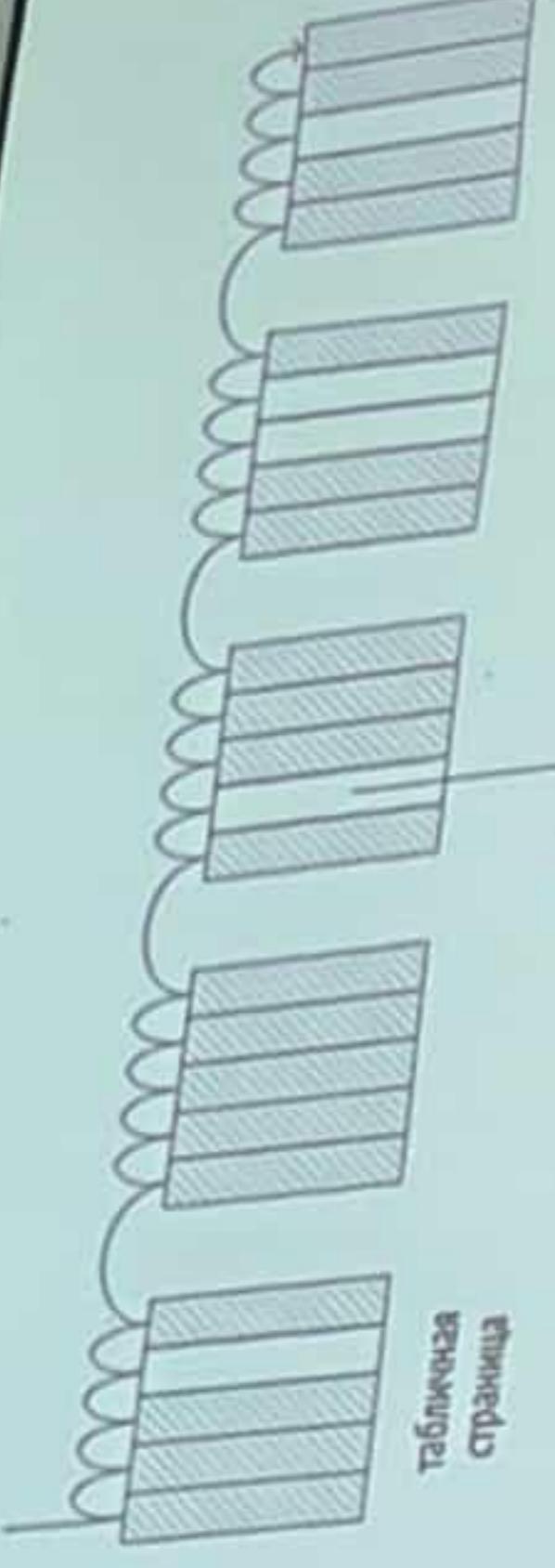


## Статистика pg\_stats

- Поле correlation представления pg\_stats показывает корреляцию между физическим расположением данных и логическим порядком в смысле операций сравнения. Если значения хранятся строго по возрастанию, корреляция будет близка к единице; если по убыванию — к минус единице. Чем более хаотично расположены данные на диске, тем ближе значение к нулю.
- Корреляция используется для оценки стоимости индексного сканирования.

## Табличные методы доступа Последовательное сканирование

- Полнотью читается файл основного слоя таблицы. На каждой странице прочитанной странице проверяется видимость каждой строки
- Чтение происходит через буферный кеш; чтобы большие таблицы не сканирующие ту же таблицу, при соединяются процессы, одновременно экономят операции дисковых чтений. Поэтому в общем случае сканирование может стартовать не с начала файла.
- Последовательное сканирование – самый эффективный способ прочитать всю таблицу или значительную ее часть. Иными словами, последовательное сканирование хорошо работает при низкой селективности.



17,6  
0,1,0,1

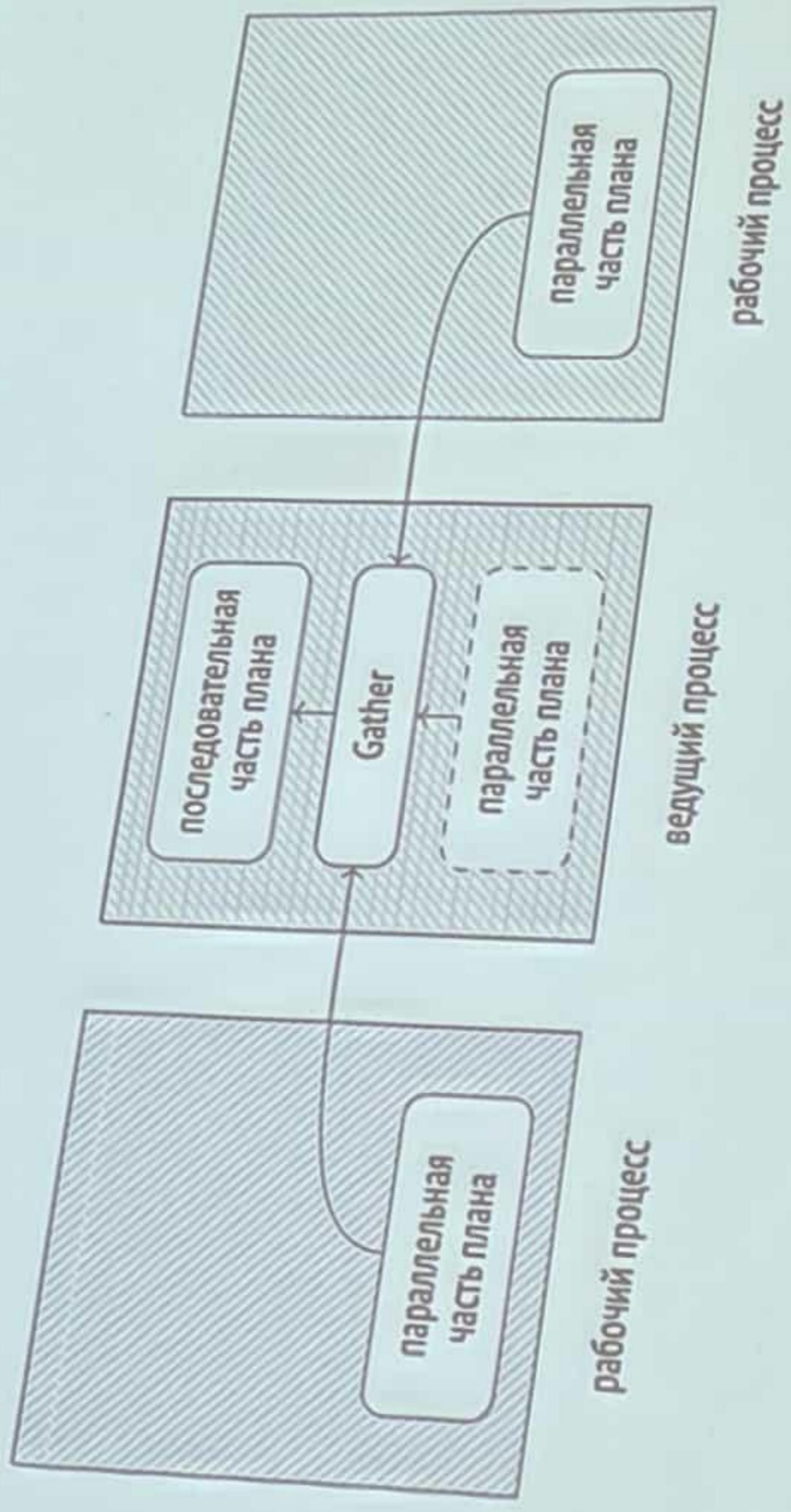
0,1,0,1

# Госледованиеательное сканирование и изъятие места пребывания

- В оценке стоимости оптимизатор учитывает две составляющие: дисковый ввод-вывод и стоимость чтения одной страницы, при условии что страницы читаются последовательно.
  - Соотношение по умолчанию подходит для HDD-дисков; для накопителей SSD имеет смысл как правило, не трогают, оставляя единицу в качестве опорного значения).
  - Оценка ресурсов процессора учитывает стоимость обработки каждой строки (которая определяется для планировщика значением параметра `seq_tuple_cost`).
  - Если на сканируемую таблицу наложены условия, они отображаются в плане запроса под узлом `Seq Scan` в секции `Filter`. Оценка числа строк будет учитывать селективность этих условий, а оценка стоимости – затраты на их вычисление.

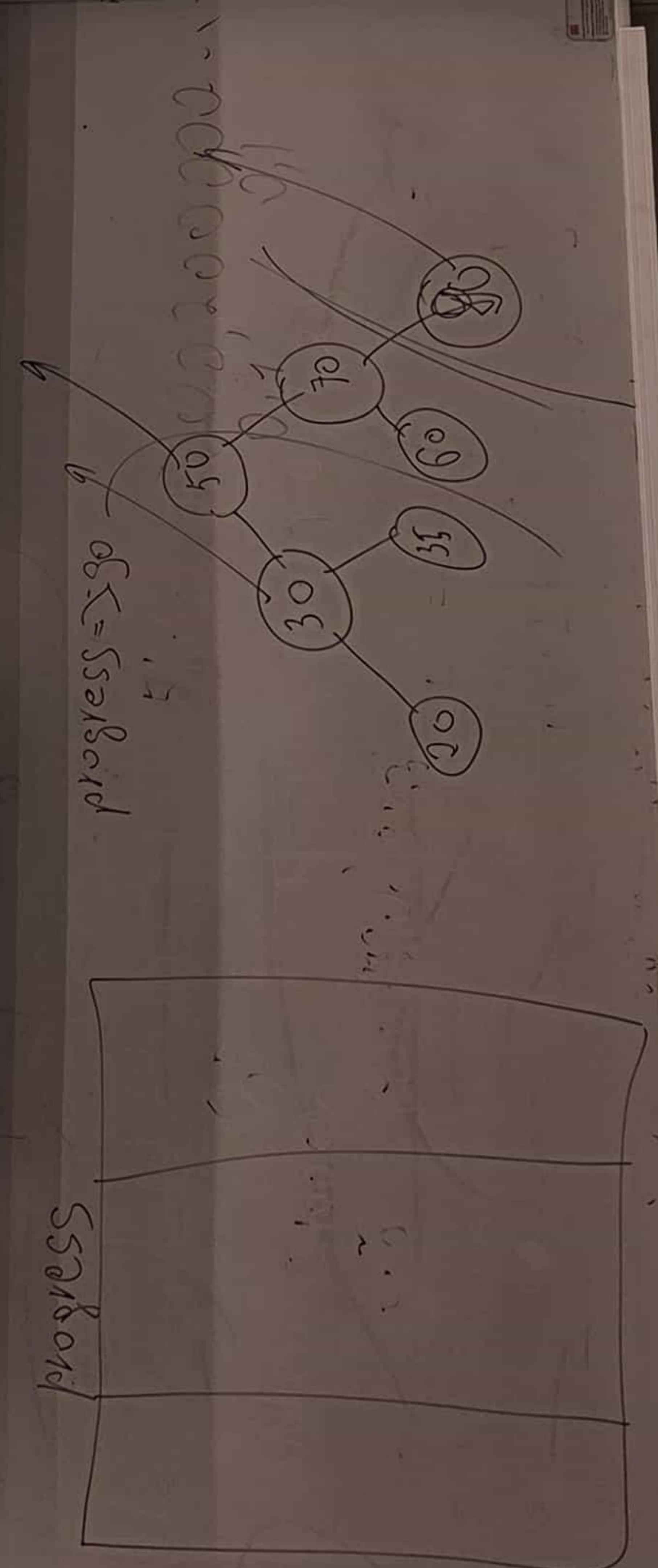
# Parallel Seq Scan – Параллельное последовательное сканирование

- Чтение выполнено последовательно специальными процессами. Процессы синхронизируются одновременно параллельно и тут же стартуют между собой участка общей памяти, чтобы помочь не пропустить**



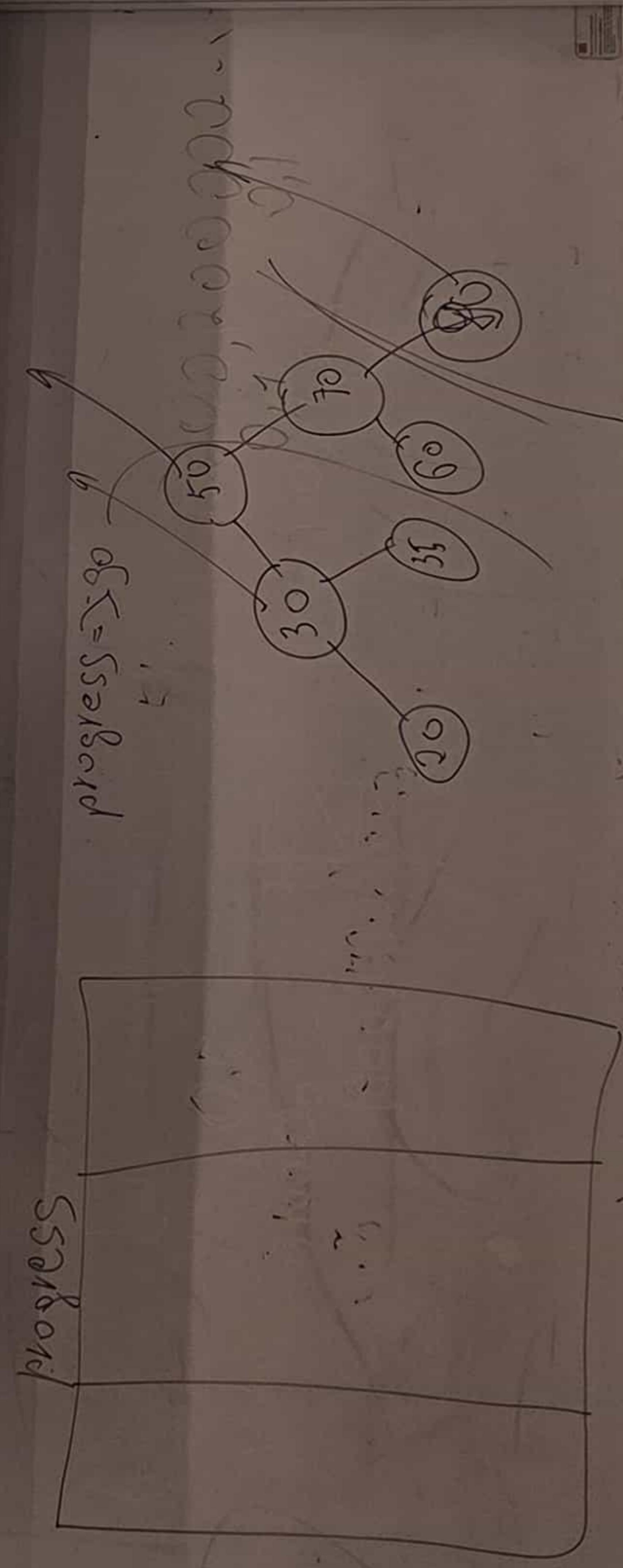
# Gather

- собирает результаты от параллельных рабочих процессов.
- Gather Merge: В этом случае параллельные рабочие процессы выполняют сортировку локально, а Gather Merge объединяет отсортированные потоки, гарантируя, что итоговый результат также этапе объединения.



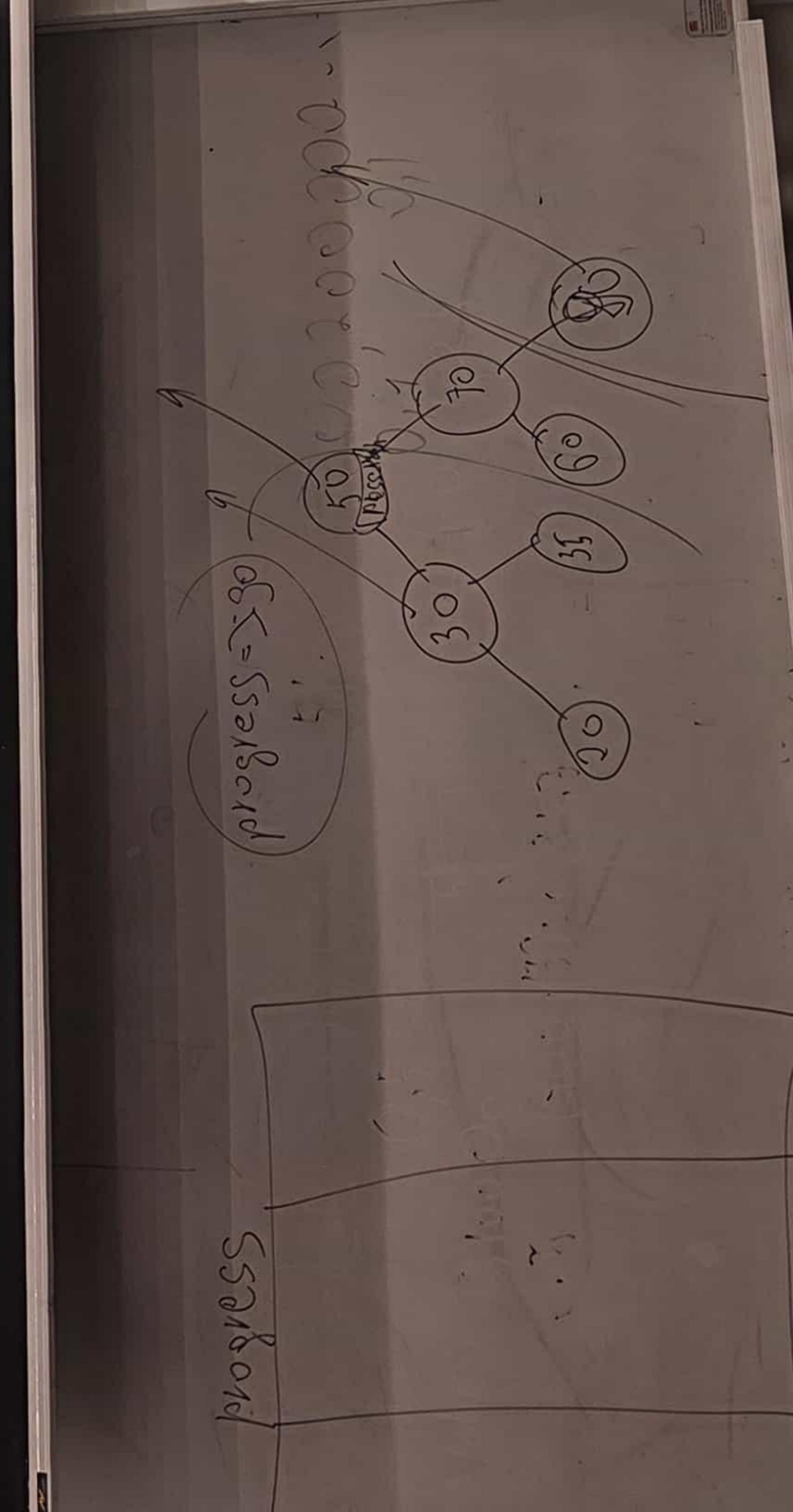
## Табличные сканирование Методы доступа

- Операция представляется в плане запроса узлом Index Only Scan страниц, сканирования только индекса, отмеченных в карте видимости.



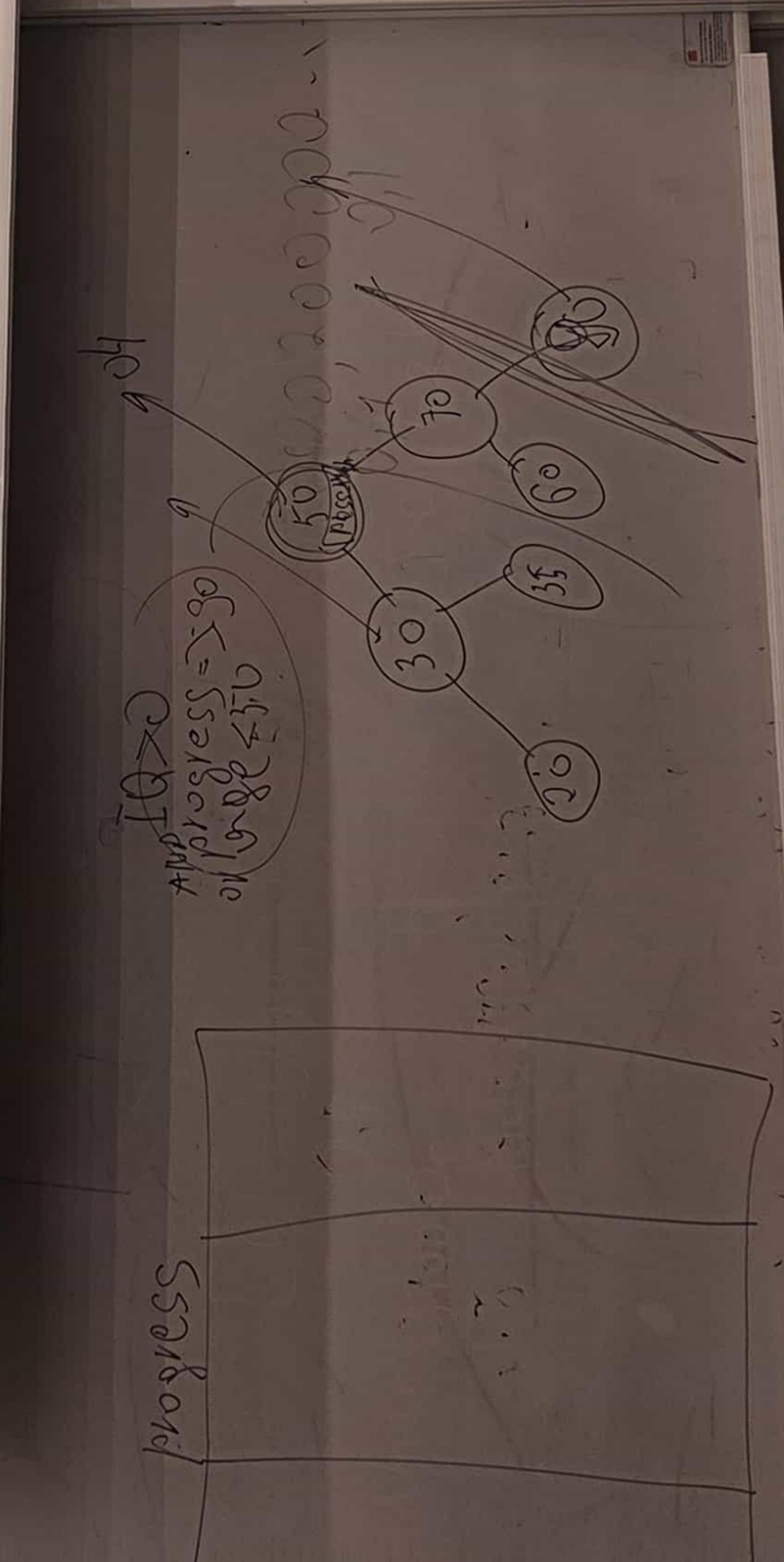
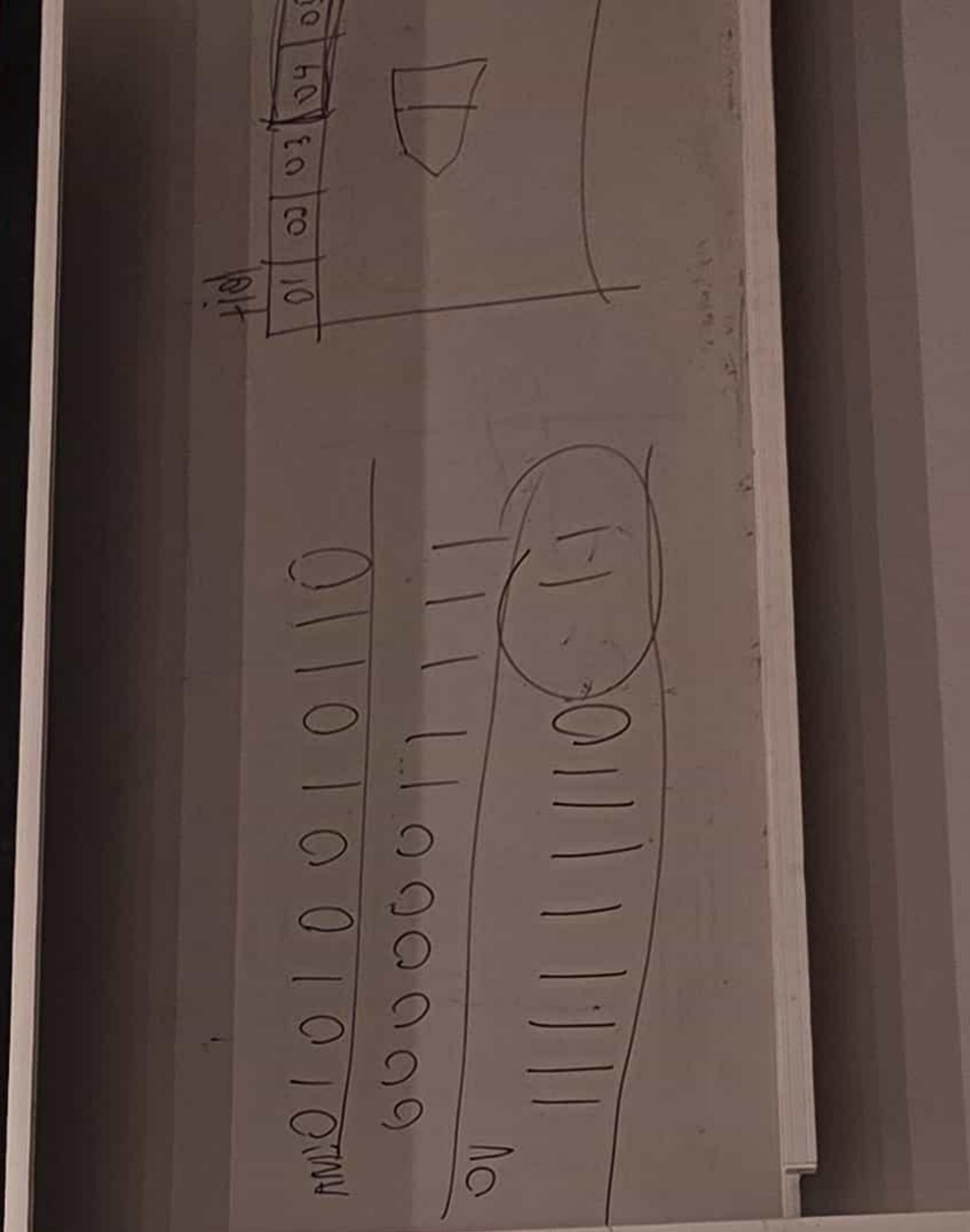
# ГЛАВНАЯ СКАДНИМРОВАНИЕ МЕТОДИКИ ПОБИТОВОЙ КАРТЕ

- Ограничение индексного сканирования связано с тем, что при корреляцией увеличивается количество странницам, а характер сбоя учащийся меняется с последовательного на




## Bitmap Index Scan

- PostgreSQL использует обычные индексы идентификаторов строк (TIDs), вместо того, чтобы сразу возврашать строки, удовлетворяющие условию запроса.
- Когда запрос использует несколько условий, можно создать битовые карты для каждого условия, а затем объединить их.



# Bitmap Near Scan

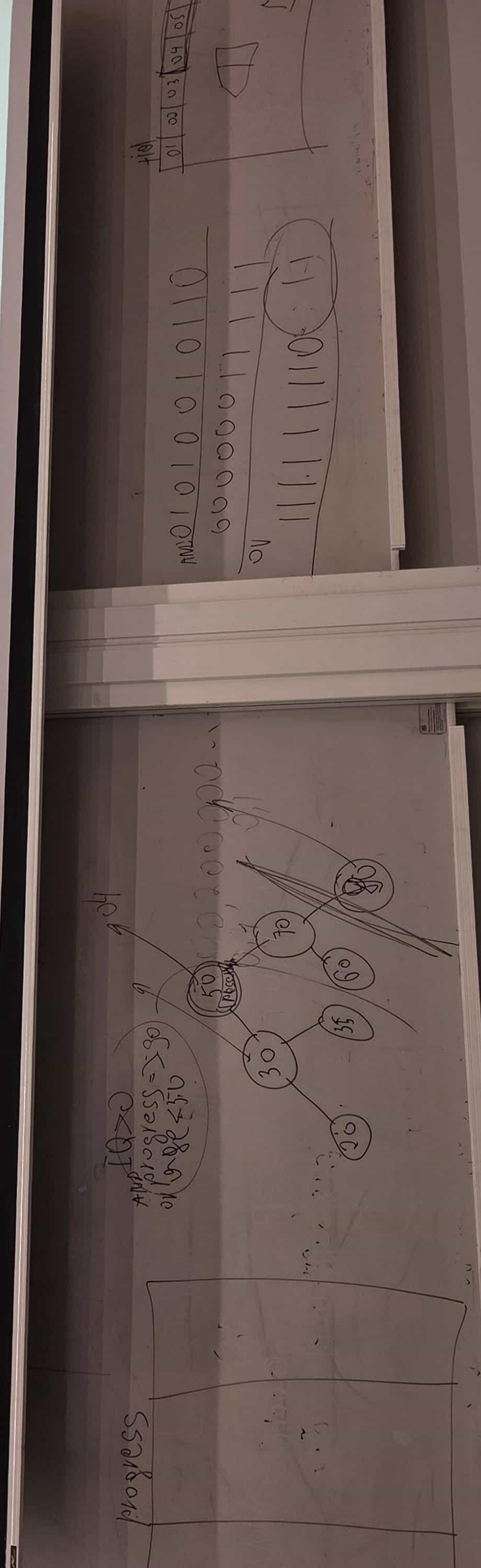
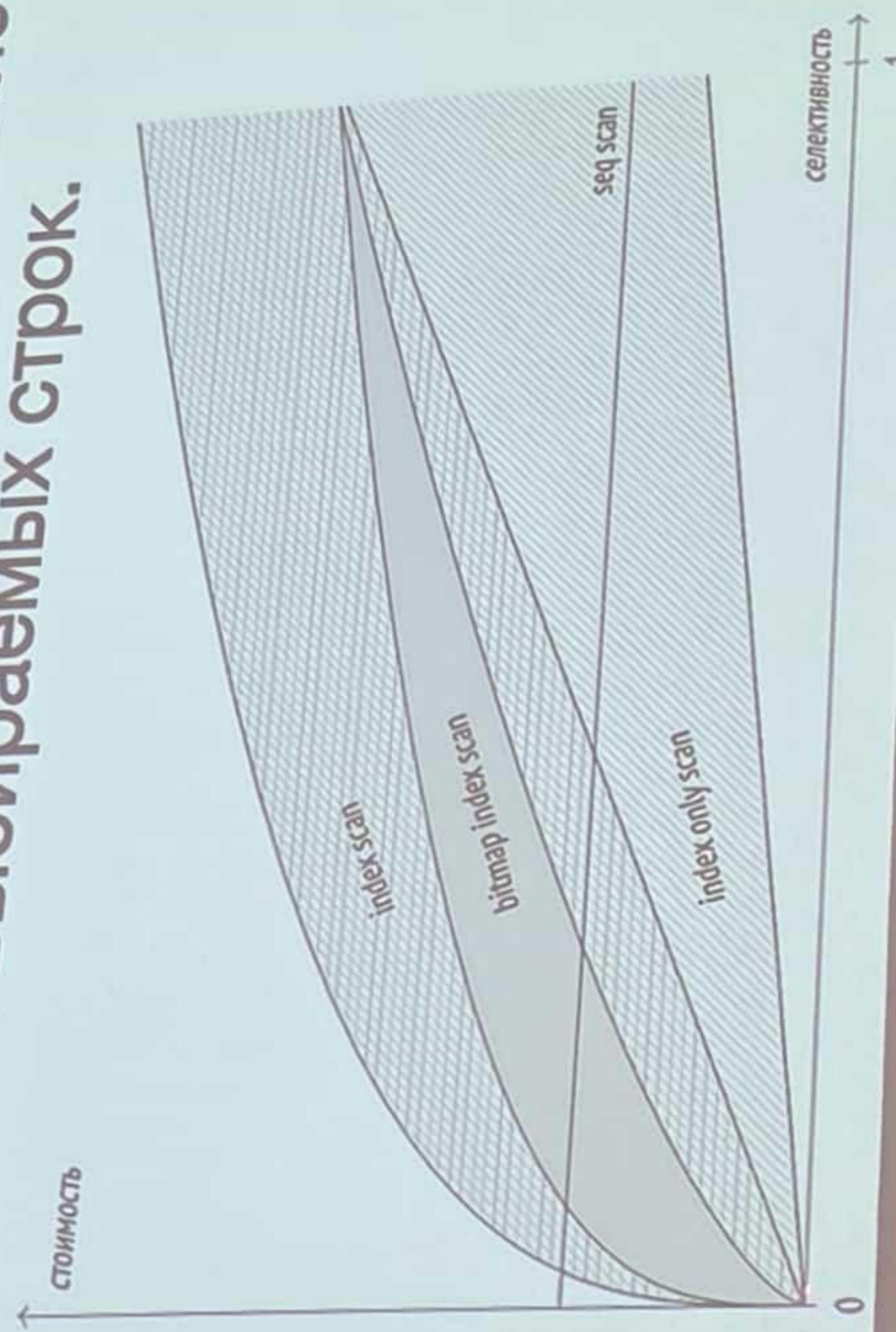
- После формирования битовой карты таблицы, не просматривая всю таблицу, осуществляется выборку строк из таблицы, не просматривая всю таблицу последовательно, а точно зная, какие строки нужно извлечь.
- Такой подход позволяет значительно сократить число операций ввода-вывода, особенно когда итоговый набор строк невелик.



# Табличные Методы

## Сравнение методов доступа

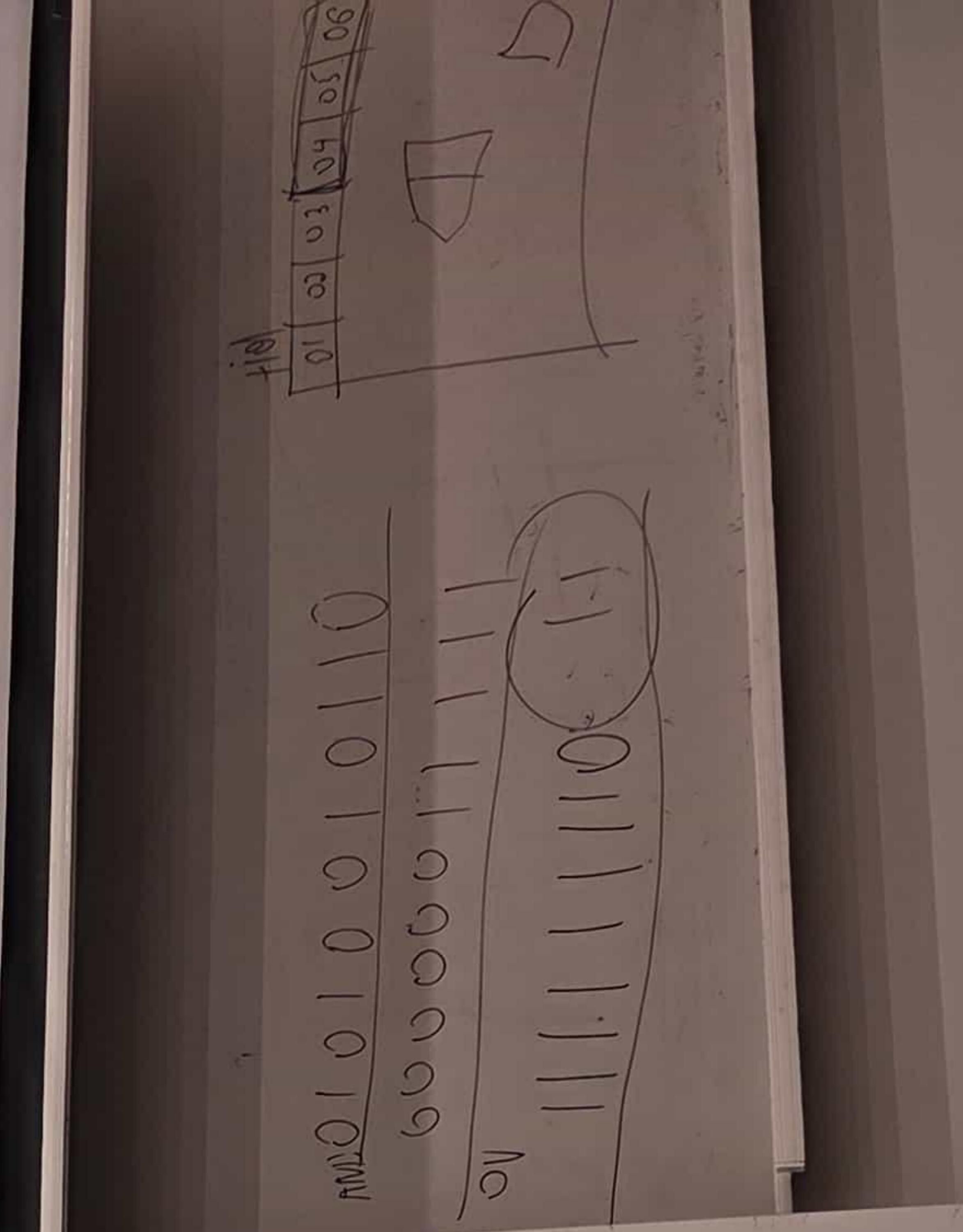
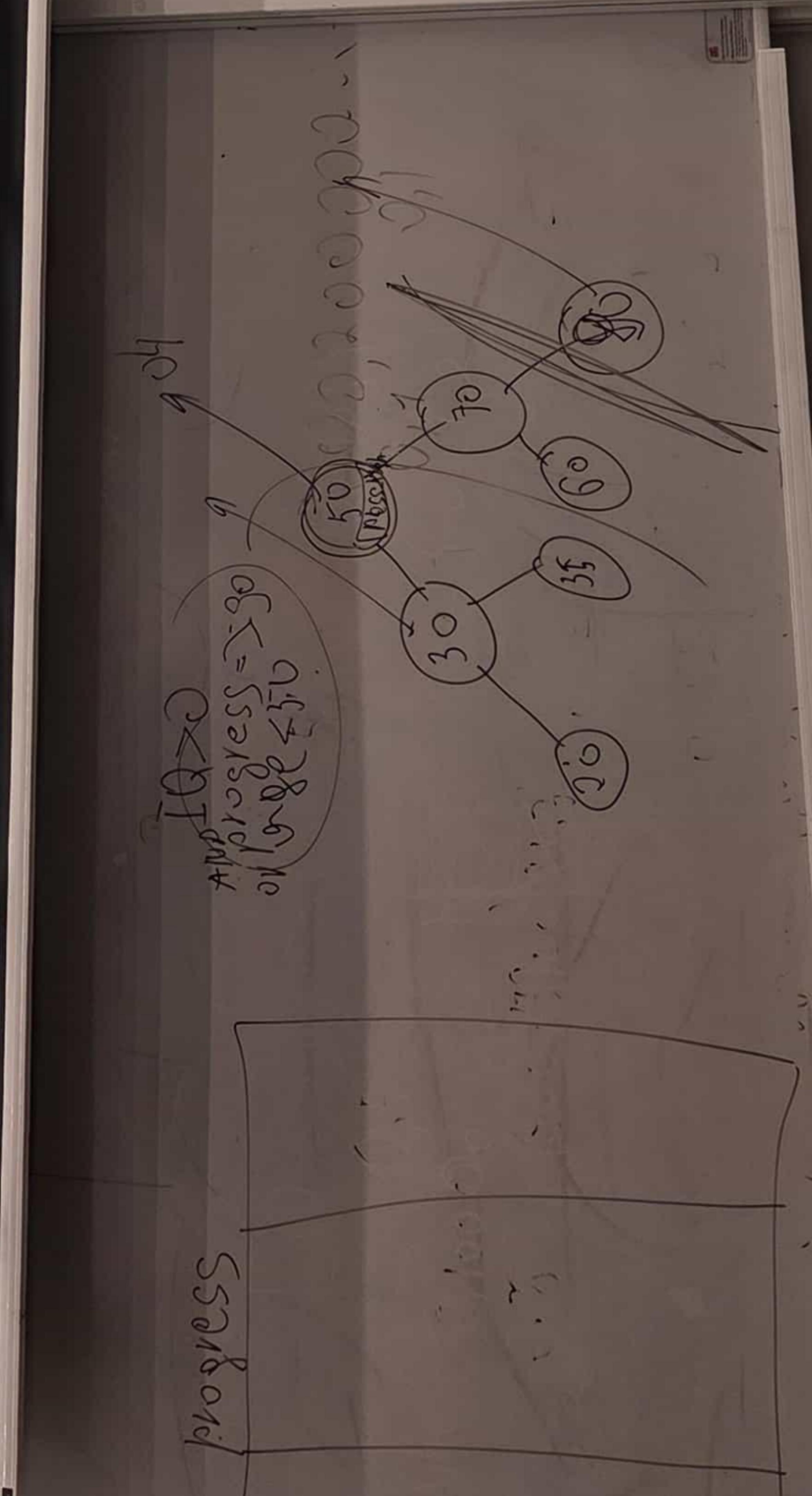
- Зависимость стоимости различных методов доступа от условий можно представить следующим образом
- Стоимость индексного сканирования сильно зависит от корреляции при идеальной корреляции индексное сканирование эффективно даже при довольно большой доле выбираемых строк.



# Виды и способы соединения вложенным циклом

- эффективность соединения вложенным циклом условий: кардинальность внешнего цикла влияет на количество строк ко внутреннему набору, позволяя использовать нескольких внутренних наборов.
- В общем случае полная стоимость соединения складывается: из стоимости первоначального получения всех строк внутреннего набора, однократной ходе которого выполняется Материализация), ( $N-1$ ) кратной стоимости каждого повторного получения строк внутреннего набора, стоимости обработки каждой строки результата.

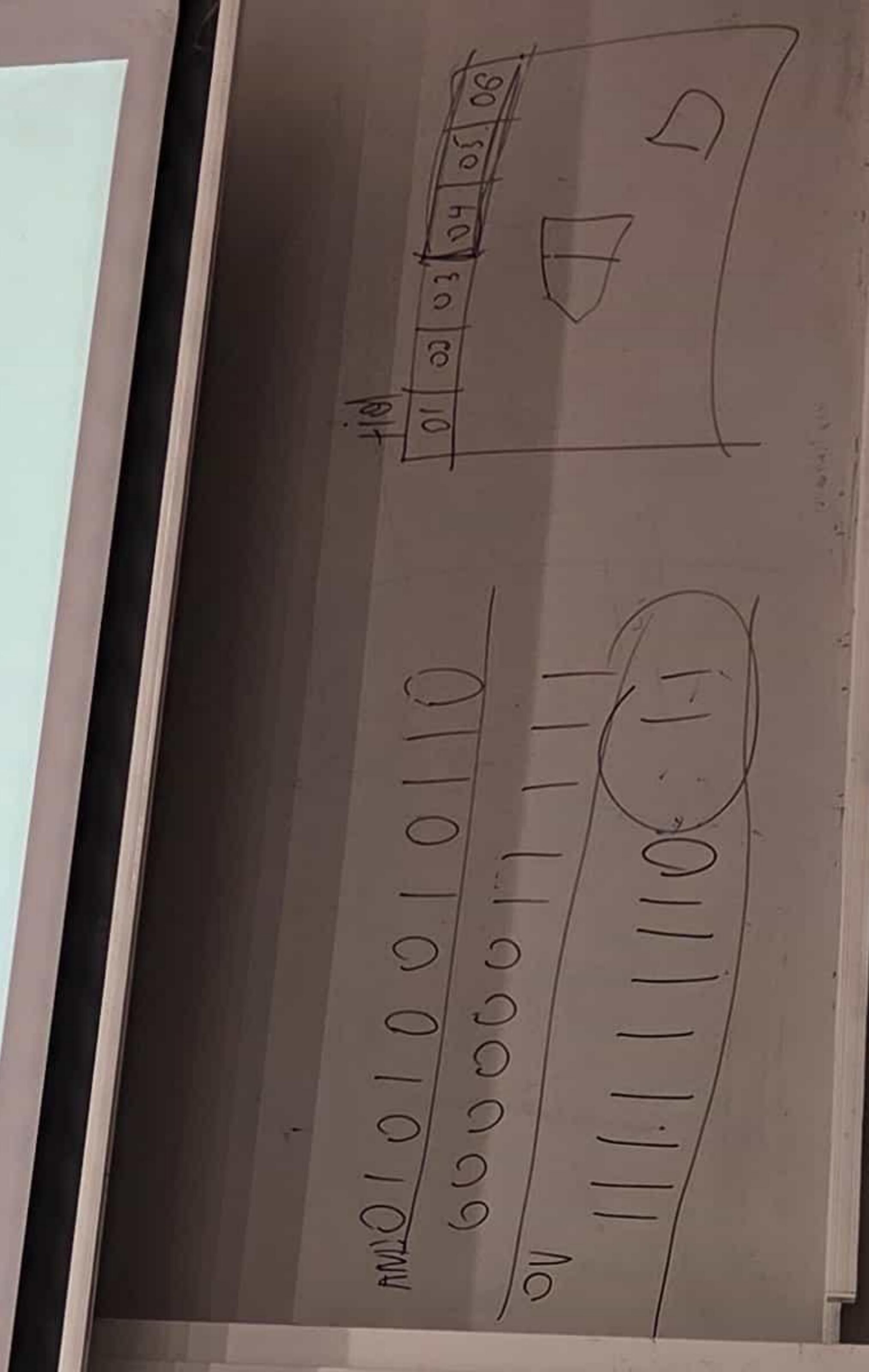
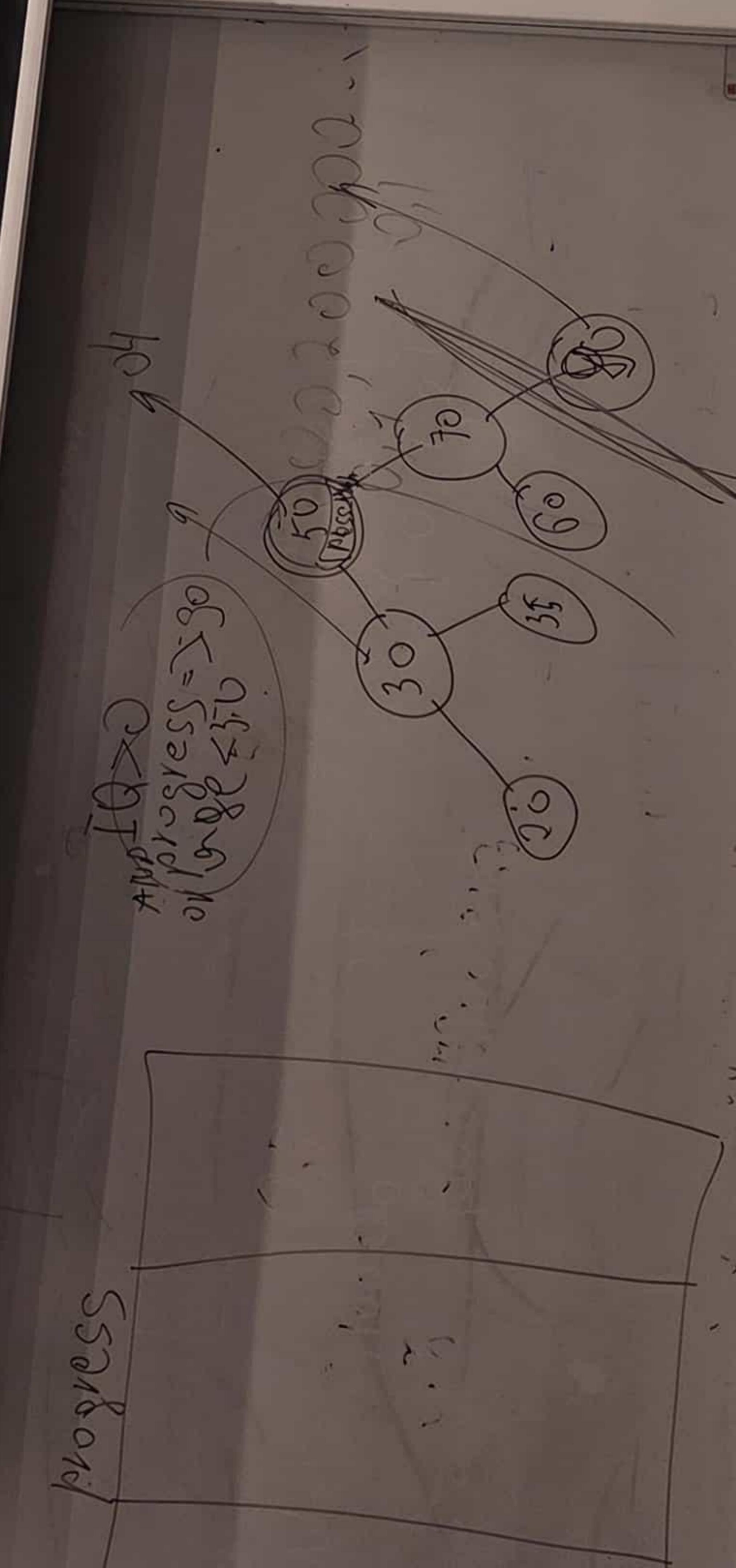
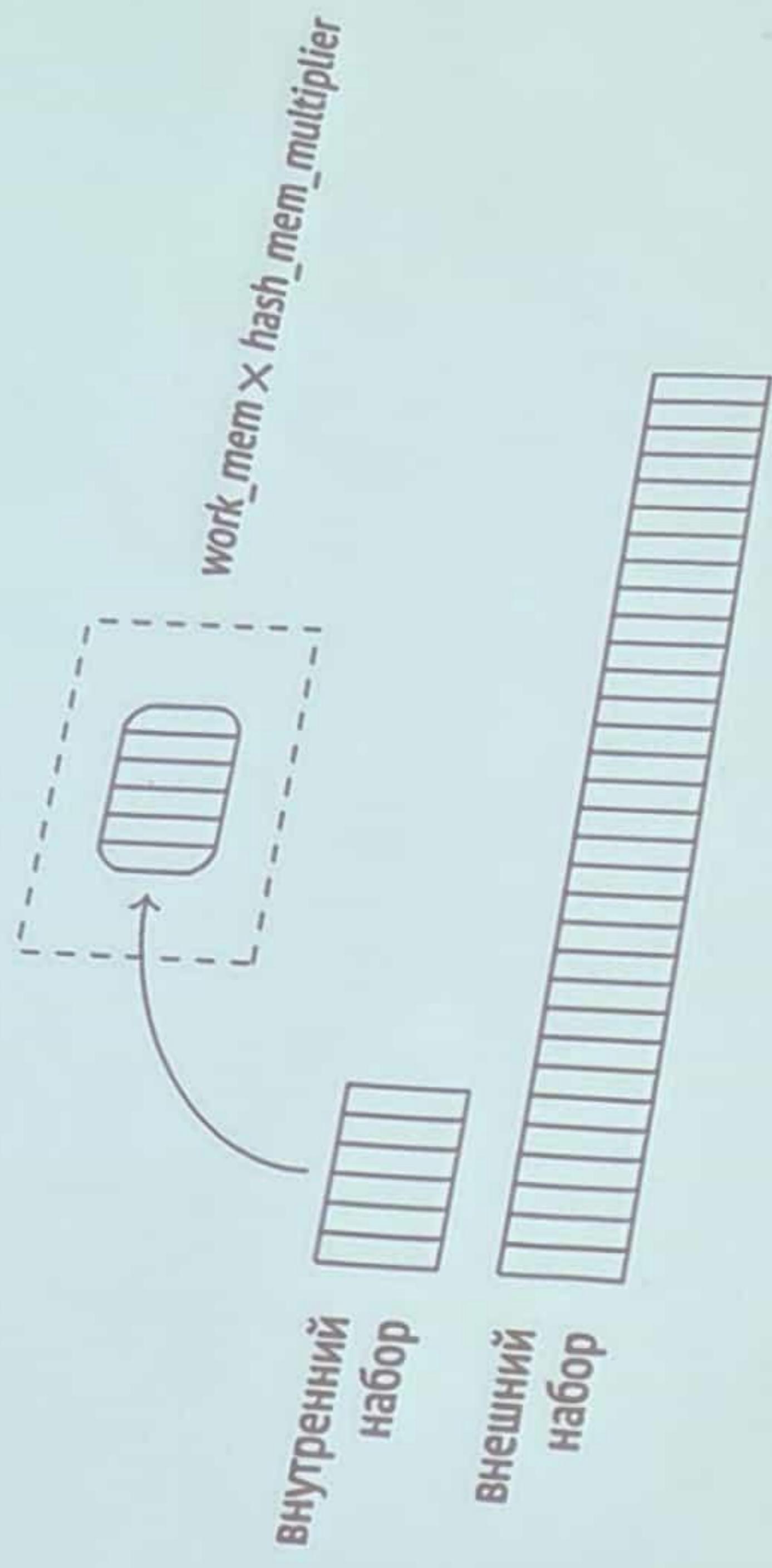
Условия: кардинальность внешнего цикла влияет на количество строк ко внутреннему набору, позволяя использовать нескольких внутренних наборов.



# Виды и способы соединения хешированием

- Реализация с разрешением коллизий

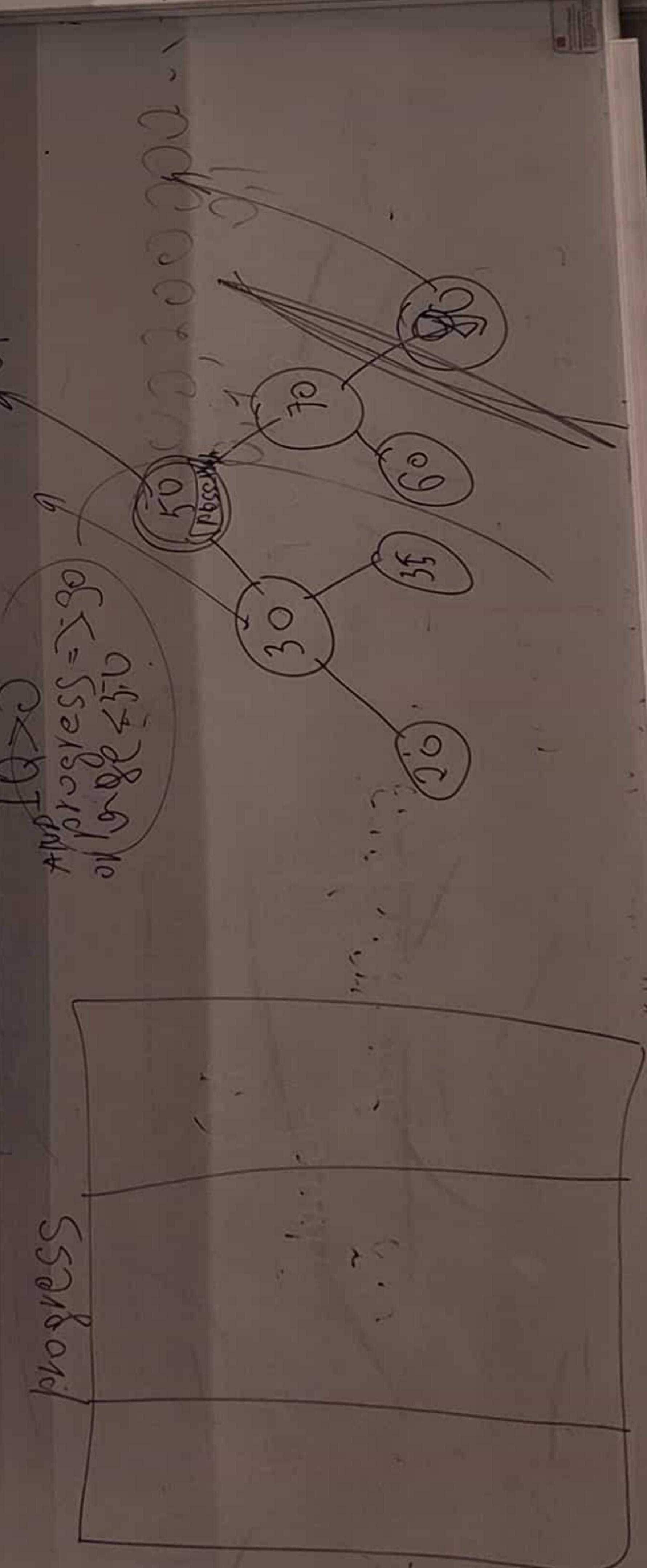
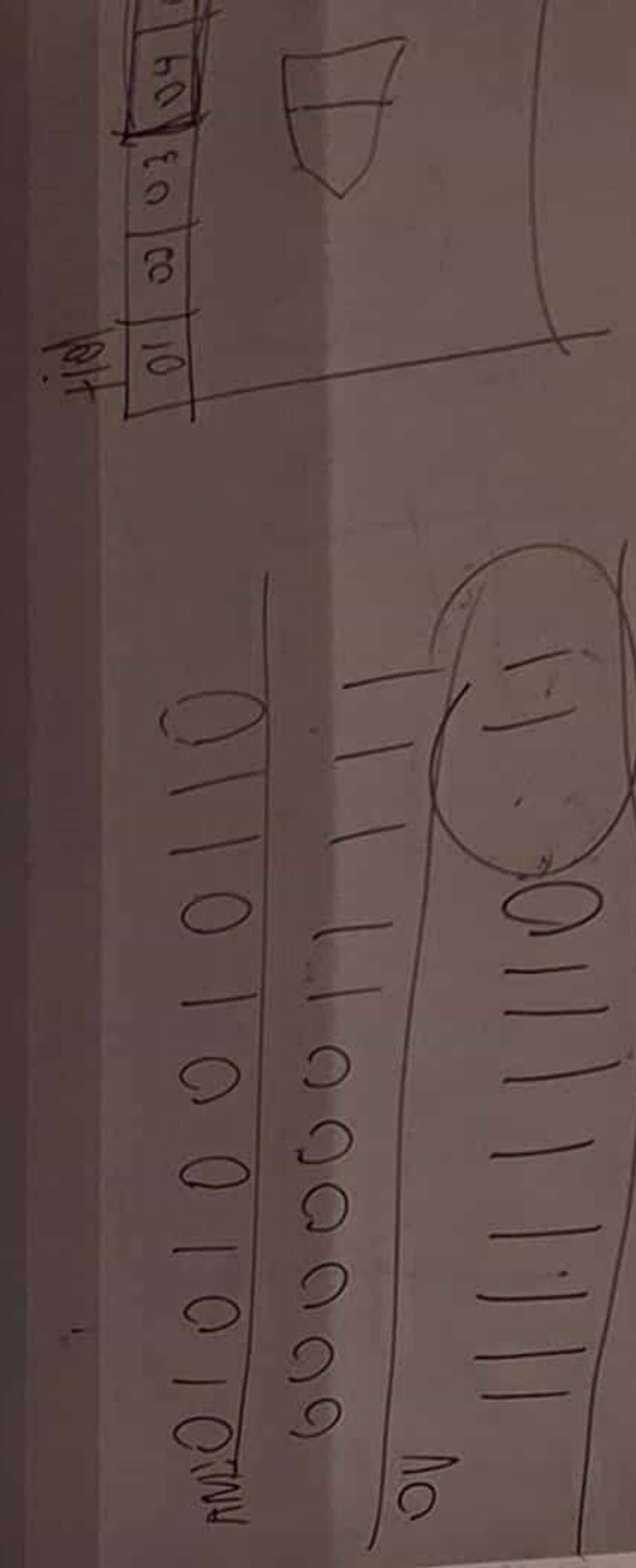
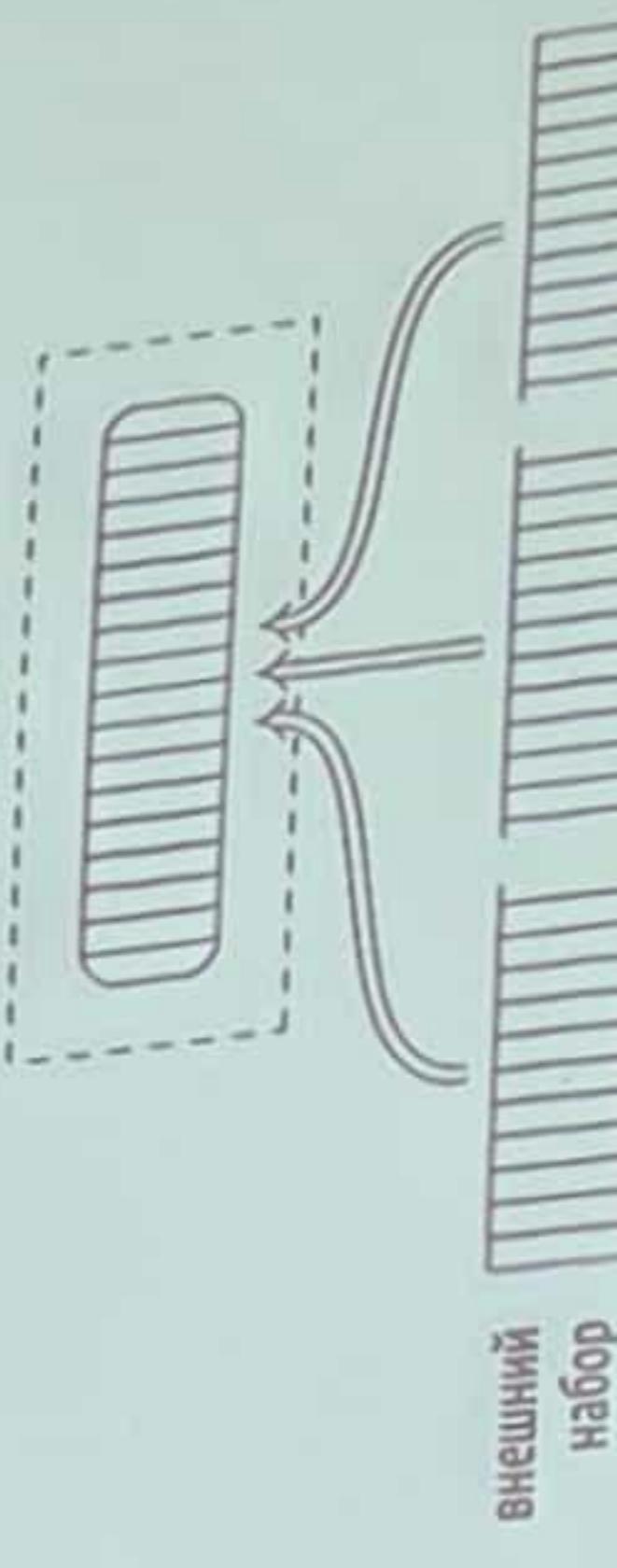
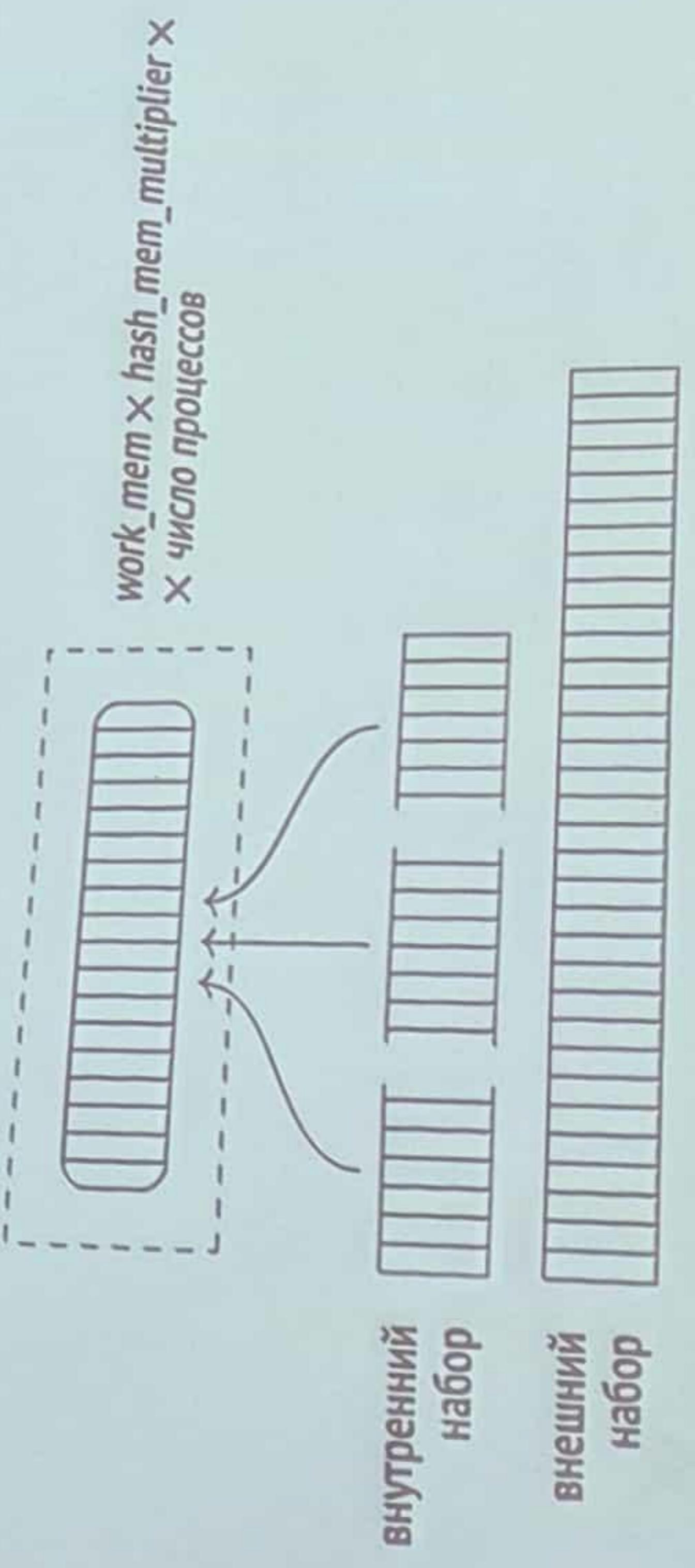
- Реализация с разрешением коллизий динамически расширяемым цепочкой, как для буферного кеша.



## Виды и способы соединений

### Соединение хешированием в параллельных планах

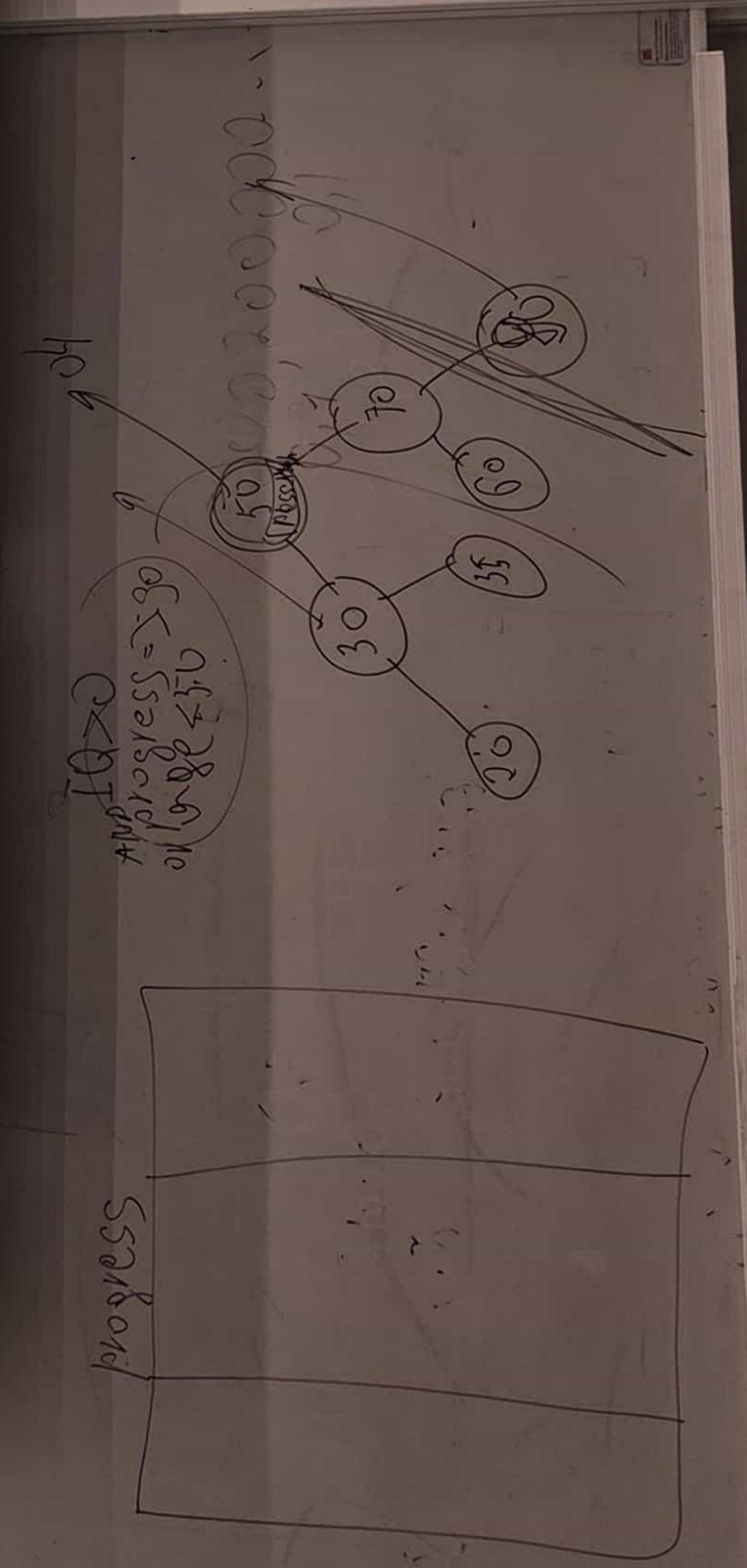
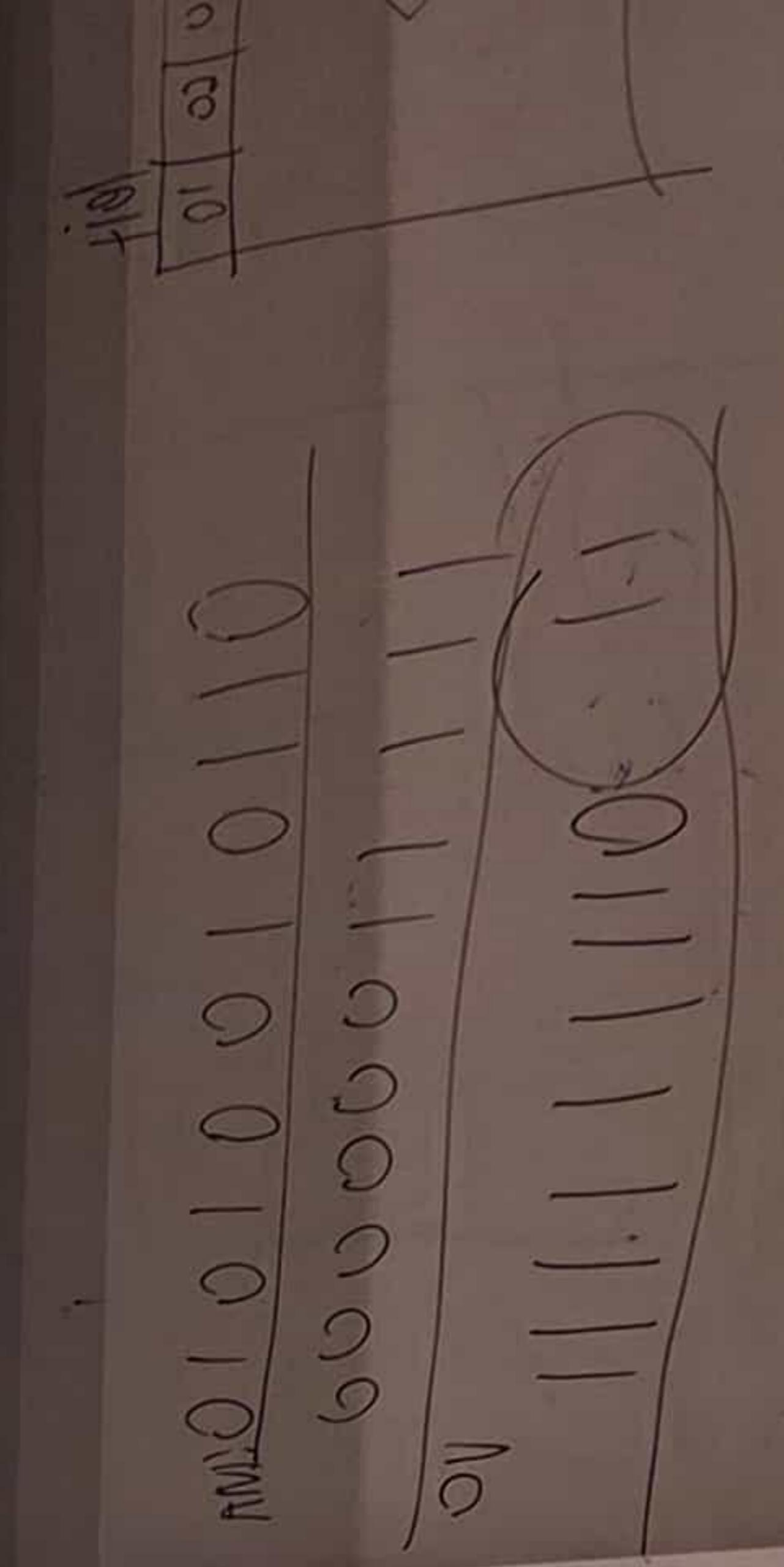
- Соединение хешированием очень эффективно для больших наборов данных. При наличии достаточно большого объема оперативной памяти оно требует однократного просмотра двух наборов данных, то есть имеет линейную сложность.



## Виды и способы соединений

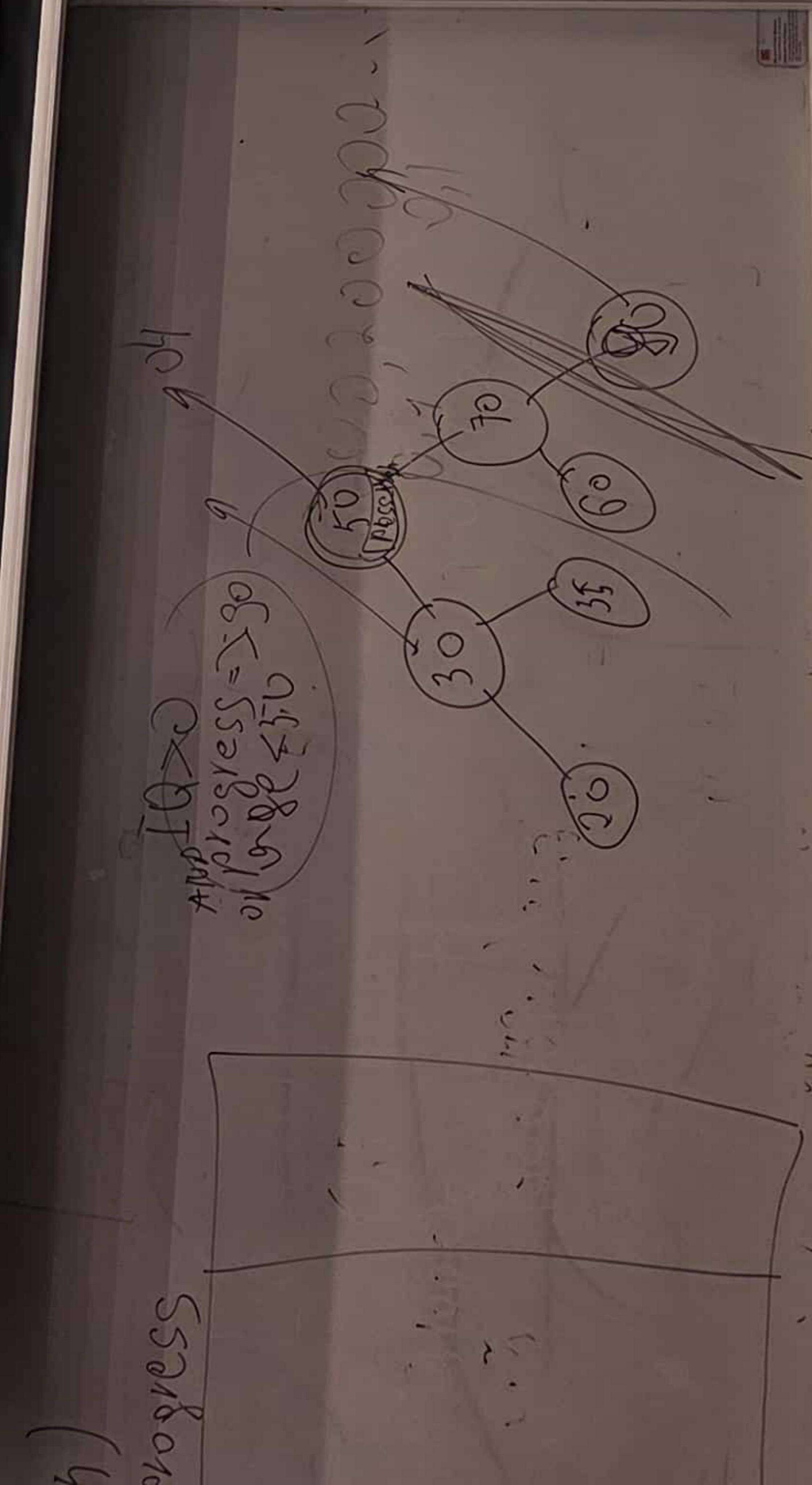
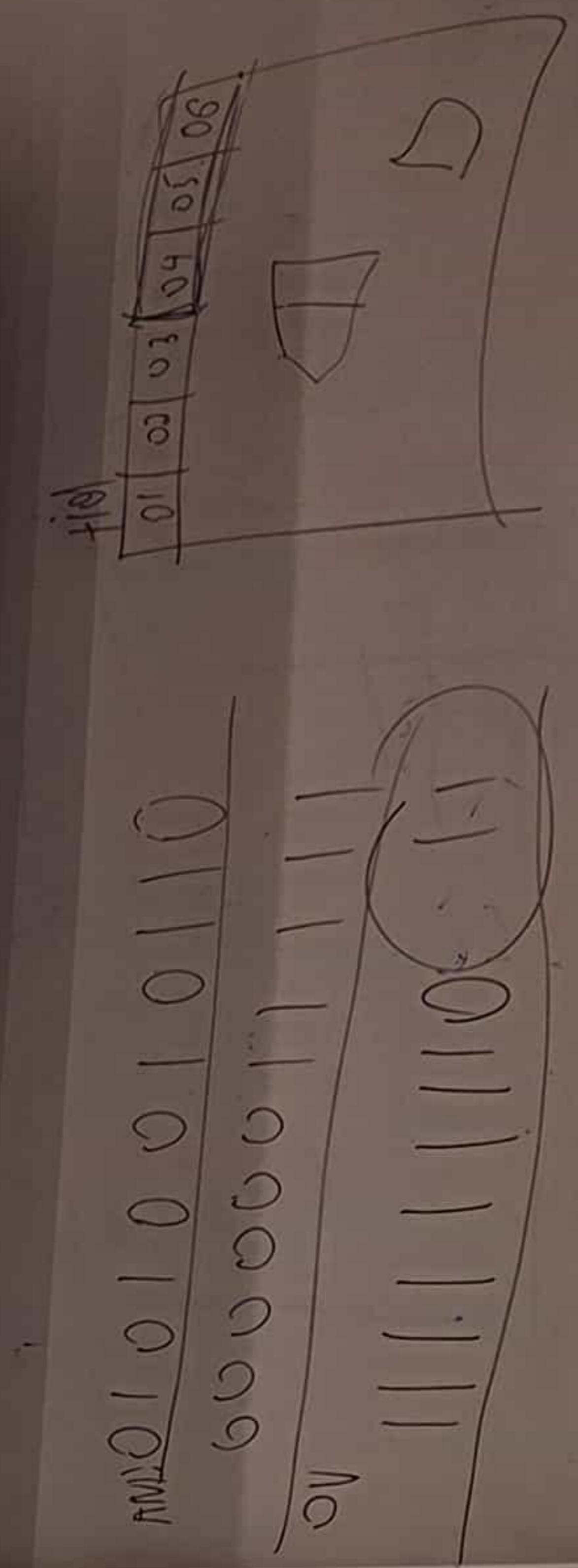
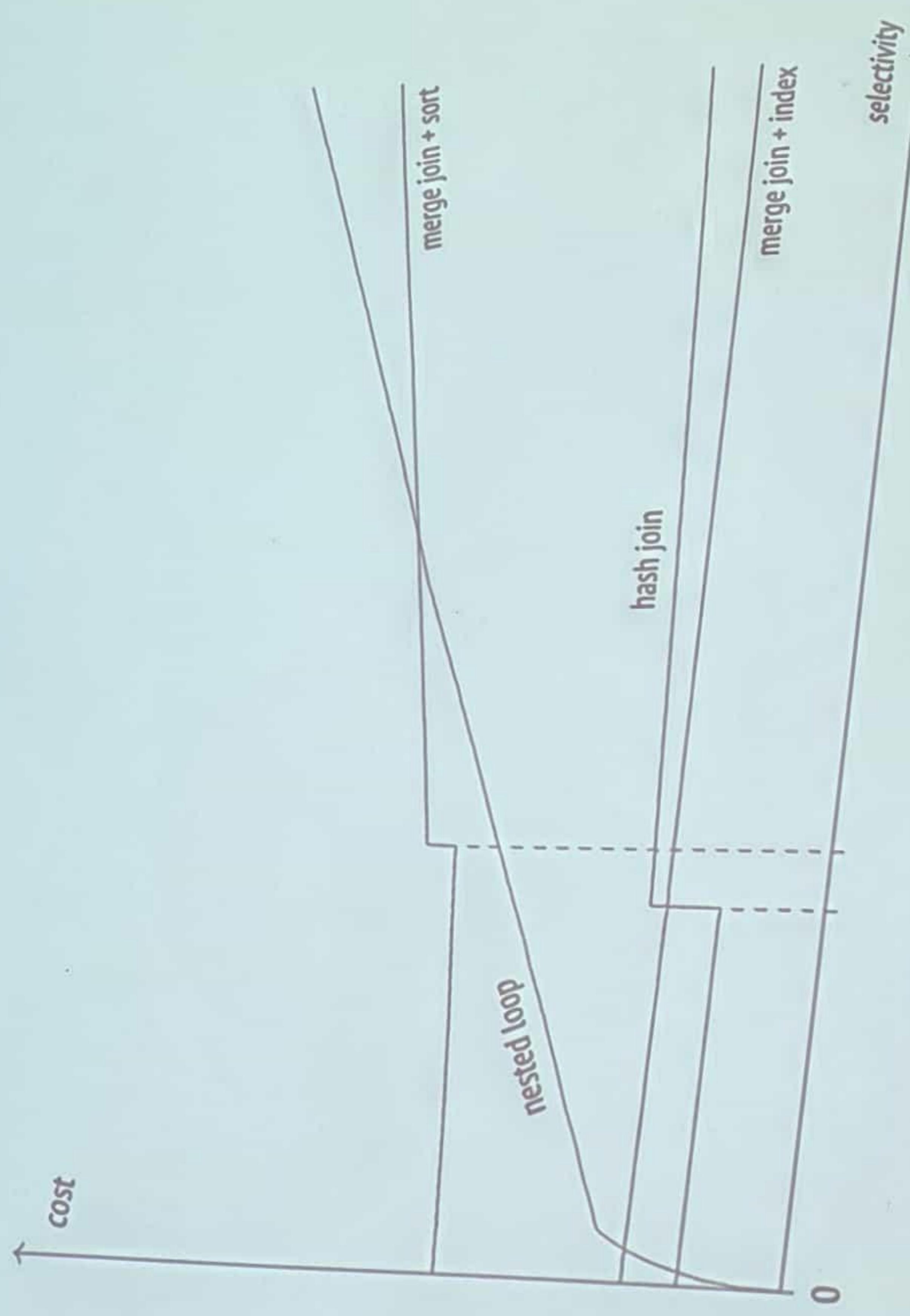
### Соединение хешированием в параллельных планах

- Соединение слиянием работает для наборов данных, отсортированных по ключу соединения, и возвращает отсортированный же результат. Входной набор может оказаться уже отсортированным в результате индексного сканирования, или он может быть отсортирован явно.
- Используется Быстрая сортировка, Частичная пирамидалная сортировка, внешняя сортировка слиянием.



# Виды и способы соединений

## Сравнение

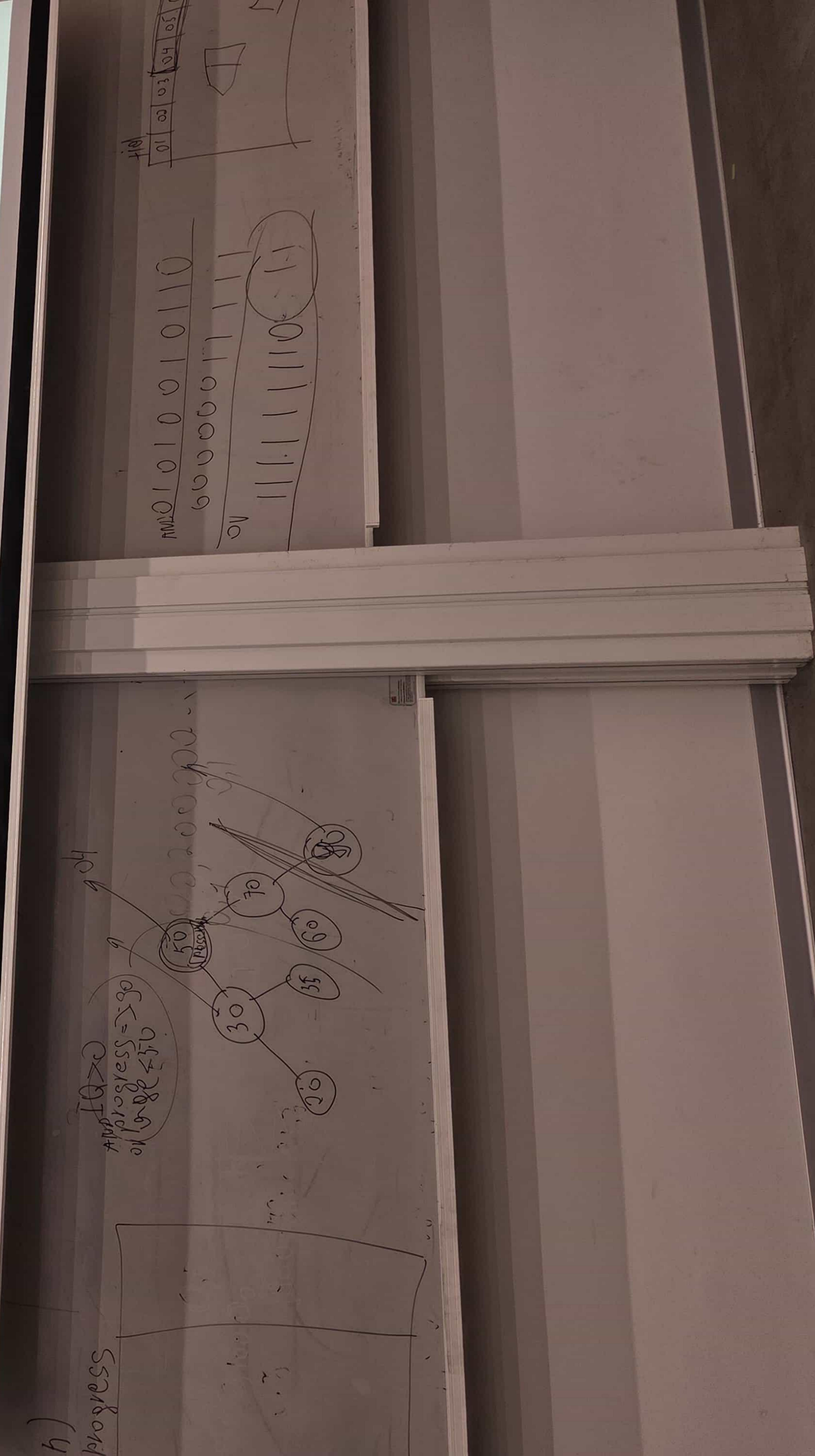


**соптимизаци**

- PostgreSQL использует quicksort, чтобы выполнить операцию в выделенной памяти. Размер выделенной памяти для сортировки контролируется параметром `work_tm`.
  - Если объем данных превышает значение `work_tm`, PostgreSQL производит сортировку с временными сохранением промежуточных результатов на диске.
  - В новых версиях PostgreSQL поддерживается параллельное выполнение сортировки.

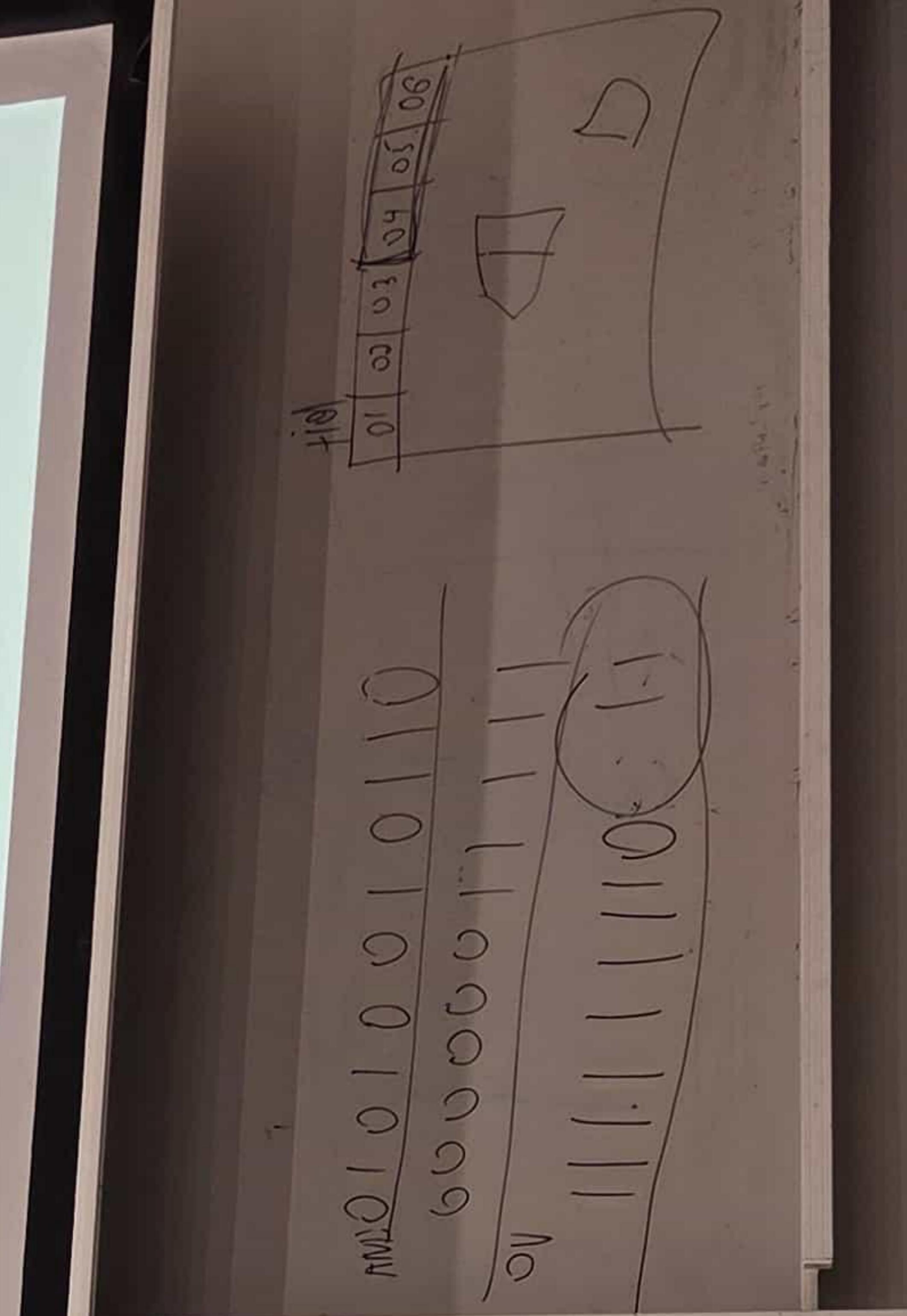
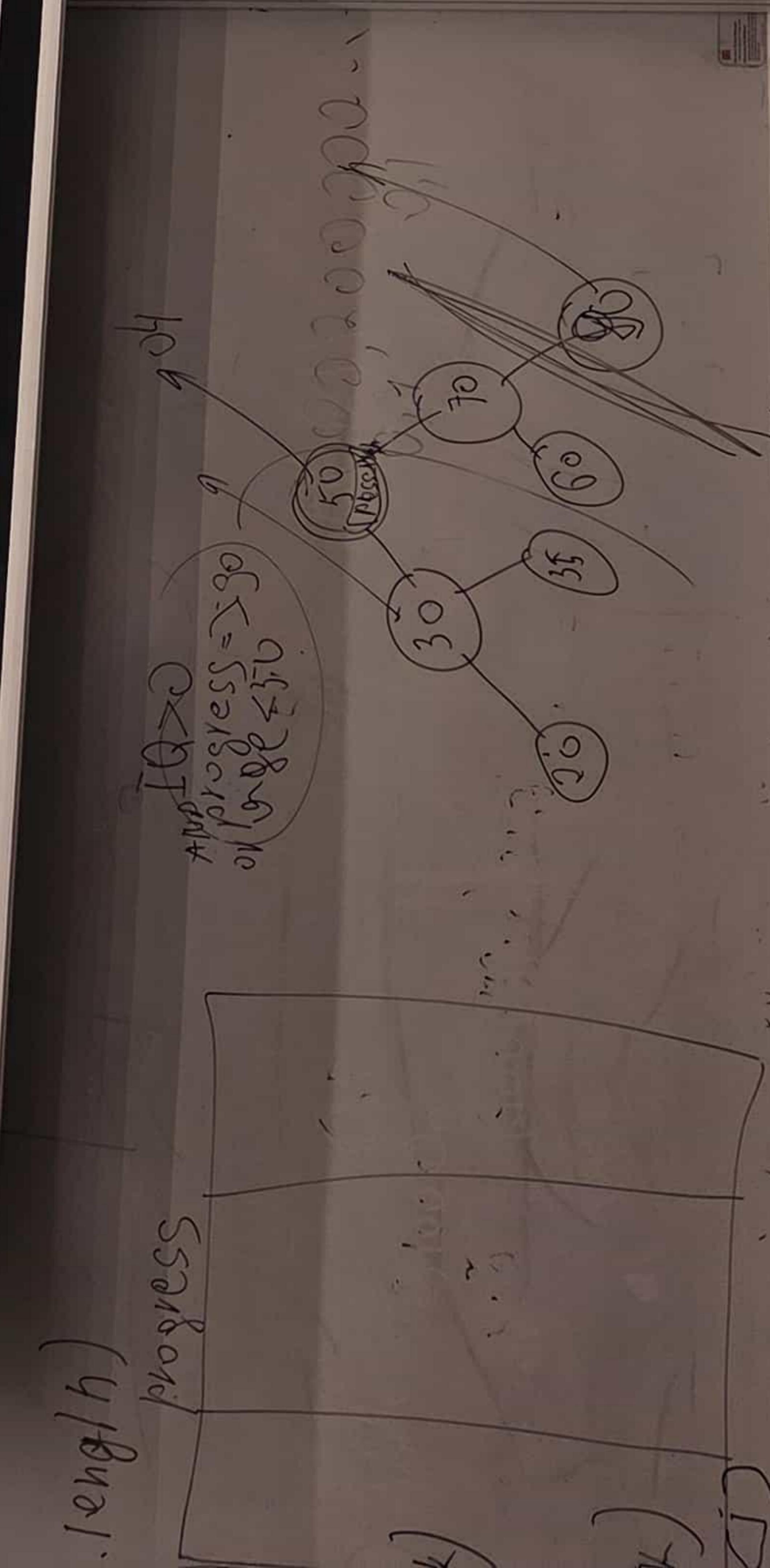
## СТЕ

- Идея была в том, что если в запросе несколько раз, то его можно определить как общее табличное выражение и ссыльаться на него столько раз, сколько потребуется. PostgreSQL вычислит результаты только один раз и повторно использует их при повторных обращениях.



## Shared buffers

- Представляют собой оперативную память, где PostgreSQL хранит копии страниц таблиц и индексов. Когда запрос кэшируется в этом кэше, чем в случае обращения к диску.



expain

- Тип операций (узлов) плана
  - Оценки стоимости (cost).  $\text{cost} = \text{START\_COST} \cdot \text{END\_COST}$
  - Размер данных (width)
  - Параллельное выполнение
  - Фактическое время выполнения. Время, затраченное на выполнение каждой операции.
  - Число циклов/итераций

