$> ls /build /data 
    /data/[filename].csv
$> echo "SELECT * FROM filename" |
    java -cp build/:jsqlparser dubstep.Main -
    red | 23 | 59.265 | pickup
    blue | 47 | 63.921 | dropout
Workflow for Script

1. data.csv
2. Filter on (rank & age)
3. Print name

Today

Last class

Optimize storage layout in general

Optimize storage layout for filter-type scripts
How do we make filter \((\text{age} > 25)\) fast?

**Idea:** Sort data by age!

1. Binary search to find \(\text{age} = 25\)
2. Scan + print all later records

**Question**
- Big blob of bits
- Where does the middle record start?
- Use fixed length records
- Create fixed size "chunks" of records
  - still need to do binary search within a chunk

**Optional:** Fixed size records

**Option 2:** Header w directory of record positions
  - (not generally used)
  - Bonus Q: Why

**Option 3:** Footer w directory of record positions
  - (actually used... why?)

**Free Bonus:** Chunk = disk page (~4kB) = cache line (64b)
Problem: Binary search \( \rightarrow \log N \) pages loaded

Idea: Summarize contents of each page & put them all on one page

Go for ages between 23 and 59, go to page 2

Problem: what if you need more than one page of summaries?

Idea: Hierarchy of summaries \( \rightarrow \) Summarize the summaries

dare I call it a "Tree"

\( \text{Pro need to load 3 pages} \)

still \( \log N \) but \( \log_k N \)

and not \( \log_2 N \)

In the language of DBs

Index: Summaries to help you get figure out where your data lives, we just invented

Trie Index: The "summary" page of "chunk"

ISAM Index: A "summary" page of chunk of records

Index page: A page on chunk of records

Data page: A page on one record
Generalizations to other filtering conditions

\[ \text{age} = X \]
\[ \rightarrow \text{binary search} \]
\[ \text{age} \geq X \quad (\text{age} \geq x) \]
\[ \rightarrow \text{binary search for page 1} \]
\[ \rightarrow \text{scan remaining} \]
\[ \text{age} < X \]
\[ \rightarrow \text{scan up to last record} \]
\[ X < \text{age} < X \]