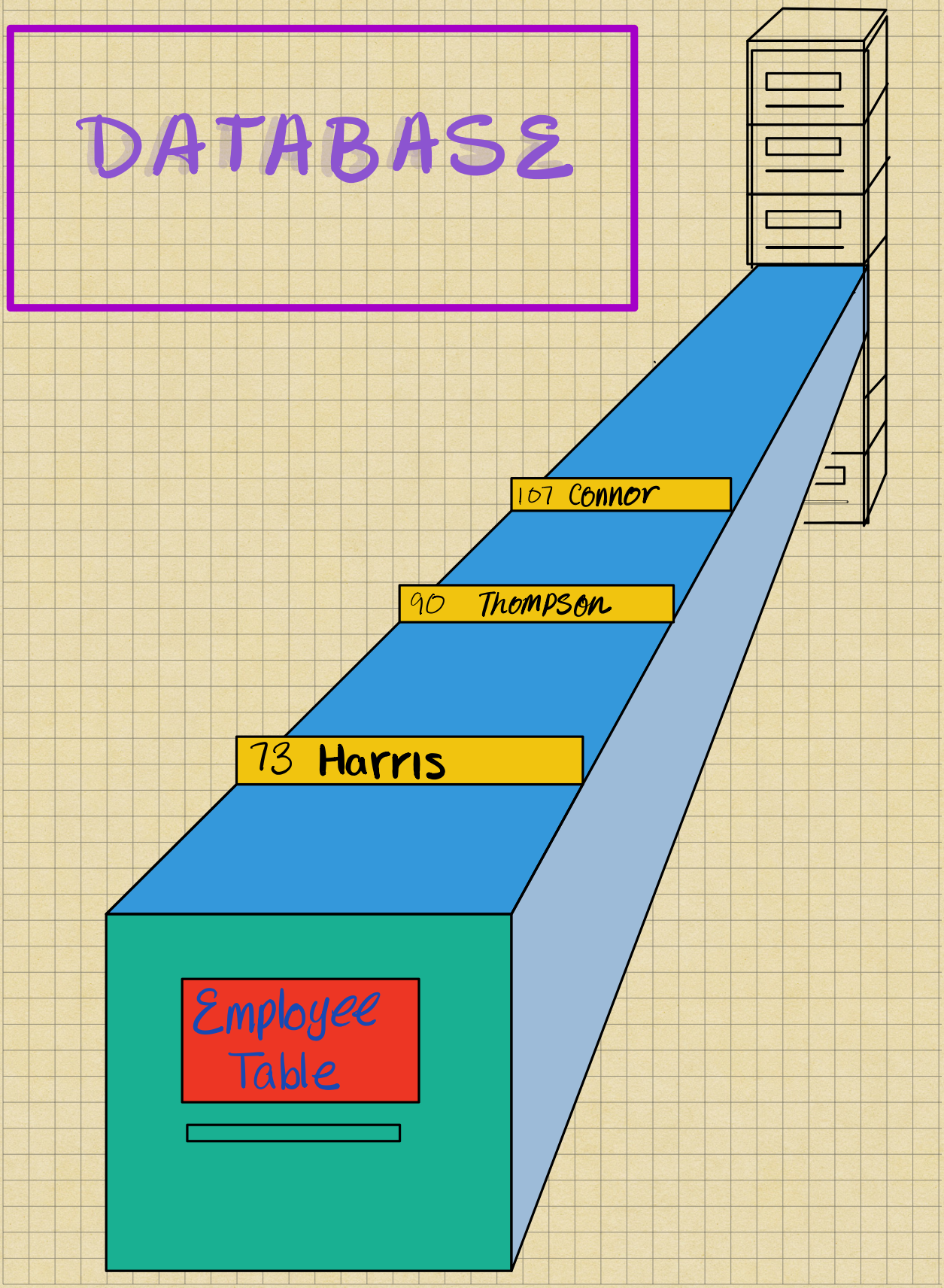

DATABASE
107 Connor
90 Thompson
73 Harris
Employee Table

Record

record no.	73
last	Harris
first	Richard
dept	Humanities
office	C 207
phone	626-585-1593

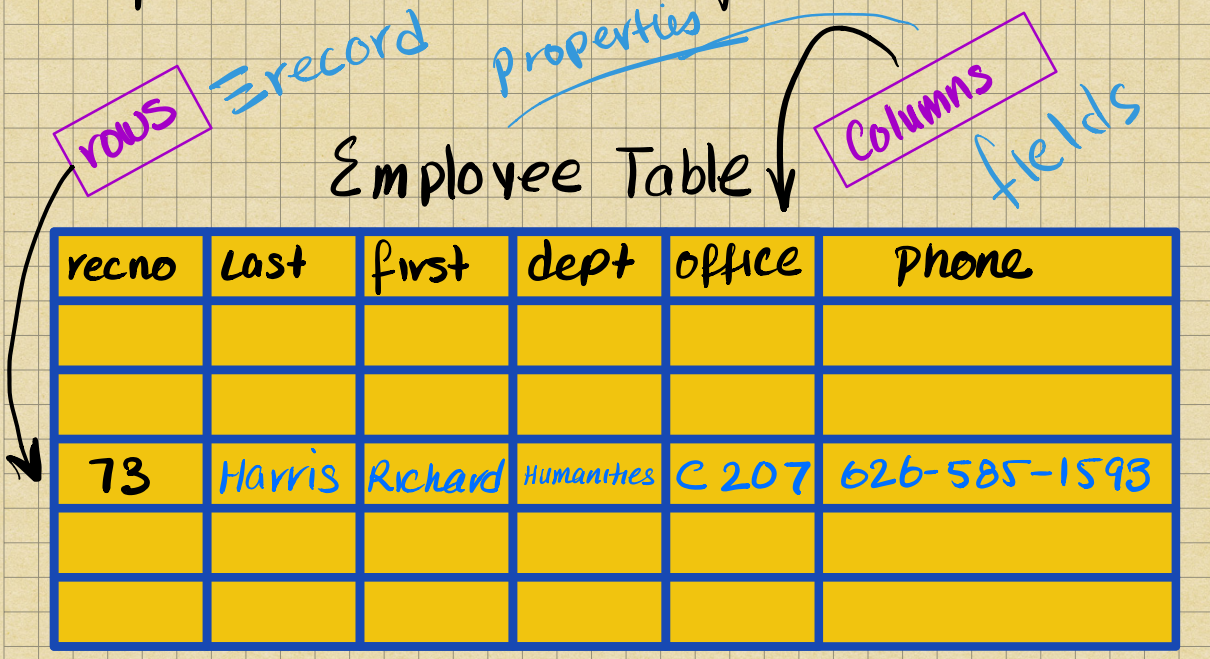
- Record: information about a single entity: one single person, place, or thing

  a collection of fields.
  each field, one single piece of info: name, address of a single person
- Record no: sequential id/pos of record in the file

  last, first, etc: fields, columns
  records: rows

Table :

- a collection of records.
- records : rows of a table
- fields : columns of a table



recno	last	first	dept	office	phone
73	Harris	Richard	Humanities	C 207	626-585-1593

!! all records in a table are the same size and have the same fields

Table class

the table class is your connection to the table on disk:

Table class:

- creates a new table: creates files, etc
- opens an existing table:
  opens files
  sets up indices and other structures
- inserts records into the table - writes them into the file
- executes a "select all" on the table returning a table object containing all the records
- executes a "select where" returning a table containing records matching one or more conditions.

- the file contains table's records in sequential chronological order.
  - each record has a record number and is stored at this position in file:
    recno * sizeof(rec)

Sample Interaction with a Table object

Construction

```
vector<string> fields ← "first", "last", "phone"
Table t("employee", fields)
```

- Give me a vector of field names
  - I will create a binary file called employee.tbl
  - write the field names into a text file called employee_fields.txt
    or write column names into the first record of the binary file after the number of fields.
    (holds the names of the cols)
  - build data structures to hold indices (vector of multimaps)
    build structures to map field names to their positional index
    fname → 0, lname → 1 etc

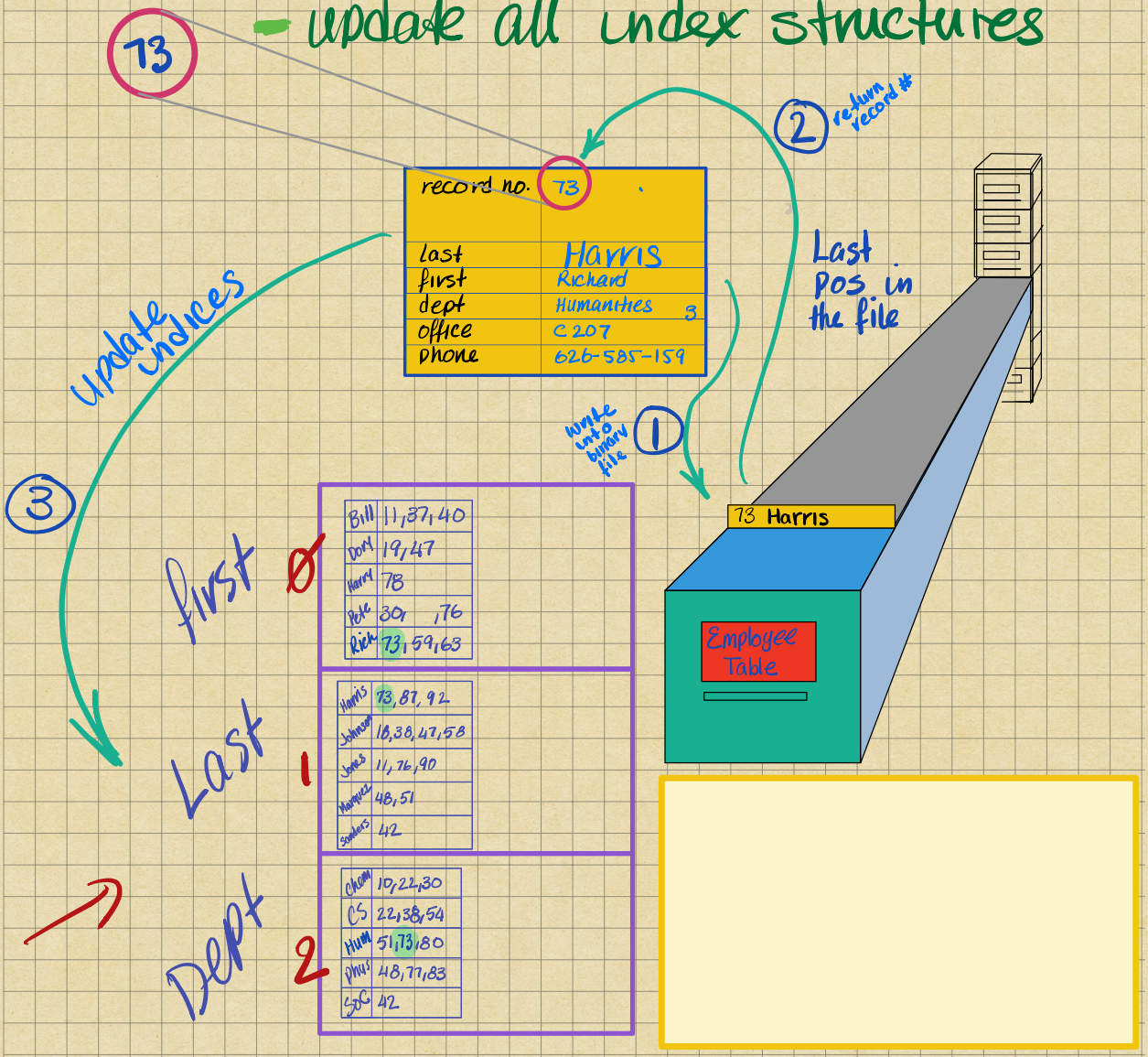
!! At no point in this project are we allowed to perform a sequential search.

Insertion of rows

t.insert_into({"Richard", "Harris", "Humanities"})

field values

- give me a vector of field values
  I will
  - write them into binary file
  - update all index structures



Data Retrieval:

Without Conditions:

```
cout << t.select_all();
```

Data Retrieval:
with conditions

```
cout << t.select(
        {"last", "first"},
    } "last", "=", "Harris"})
```

columns

rows

condition:

Last	=	Harris

field_map

first	0
Last	1
Dept	2

indices[1][Harris]

indices

0 first

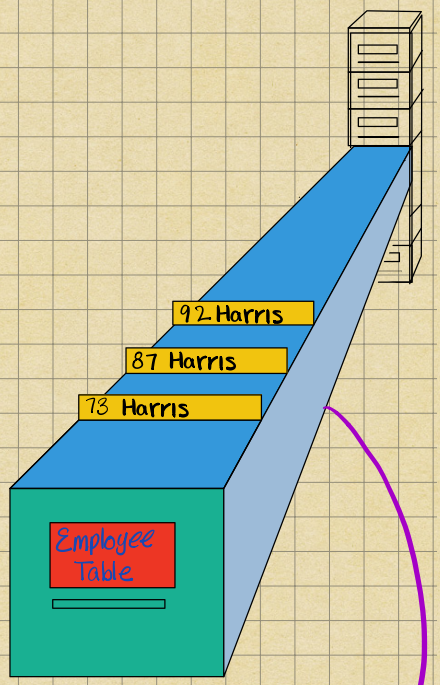
Bill	11,37,40
Dory	19,47
Harry	73
Pete	30,73,76
Rich	73,59,63

1 Last

Harris	73,87,92
Johnson	18,38,47,58
Jones	11,76,90
Marquez	48,51
Sanders	42

2 Dept

Chem	10,22,30
CS	22,38,54
Hum	51,73,80
Phys	48,77,83
Soc	42



Harris	73,87,92

r=

record no. 92
last Harris

record no. 87
last Harris

record no. 73

last	Harris
first	Richard
dept	Humanities
office	C207
phone	626-585-1593

Table Member Vars & functions:

- table name
- indices vector and/or map
  vector of all index structures for this table
- field names vector: field names
- map of field names & their order:
  fname:0, lname:1, dept:2 ...
- bool empty: no records have been added
- serial: Global serial number
- last_record number

- Table CTORS
- insert_into
- select all
- select

- setfields (vector of field names)
- get_fields returns vector of field_names

- reindex

TO DO:
set field_names
set_name
last_record=0
serial++
create an empty file
write_info table's field_list+name
? build keyword list


CTOKS
open or create
a table

- Table (const string & name, const vector<string> fields)

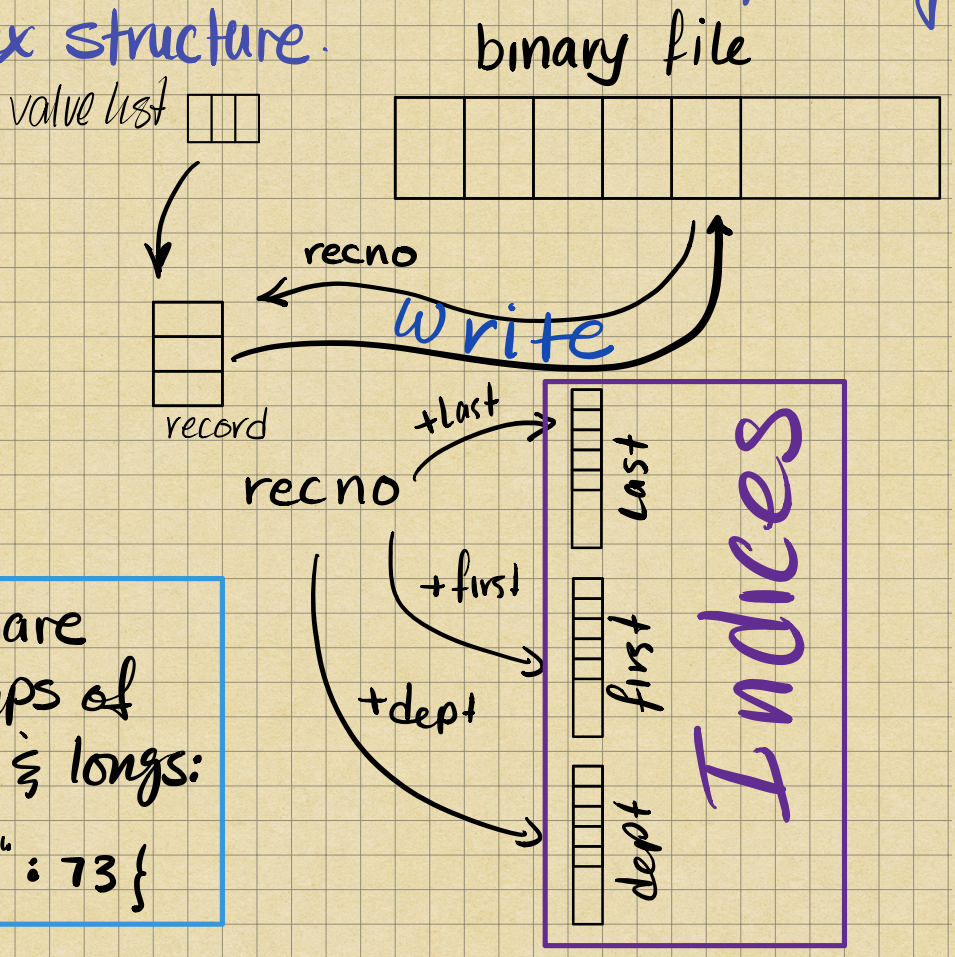
  creates a new table. (table does not already exist)
  - senal++
  - set_fields to this vector (fields)
    - build:
      - indices vector
      - field map
      - field names vecter
  - create an empty binary file with same name as table
  - write_info: write the names of the fields to text file: tablename_fields.txt
    text file ← vector of field names
  - Build index structures
  - Build various lookup maps

- Table(const string& name)
  open an existing table.
  - open / build index structures
  - open filestream
  - read_info: read field names from
    - a text file: textfile → vector
      set_fields to this vector:
      see above
  - reindex
    (re)build index structures:

insert_into

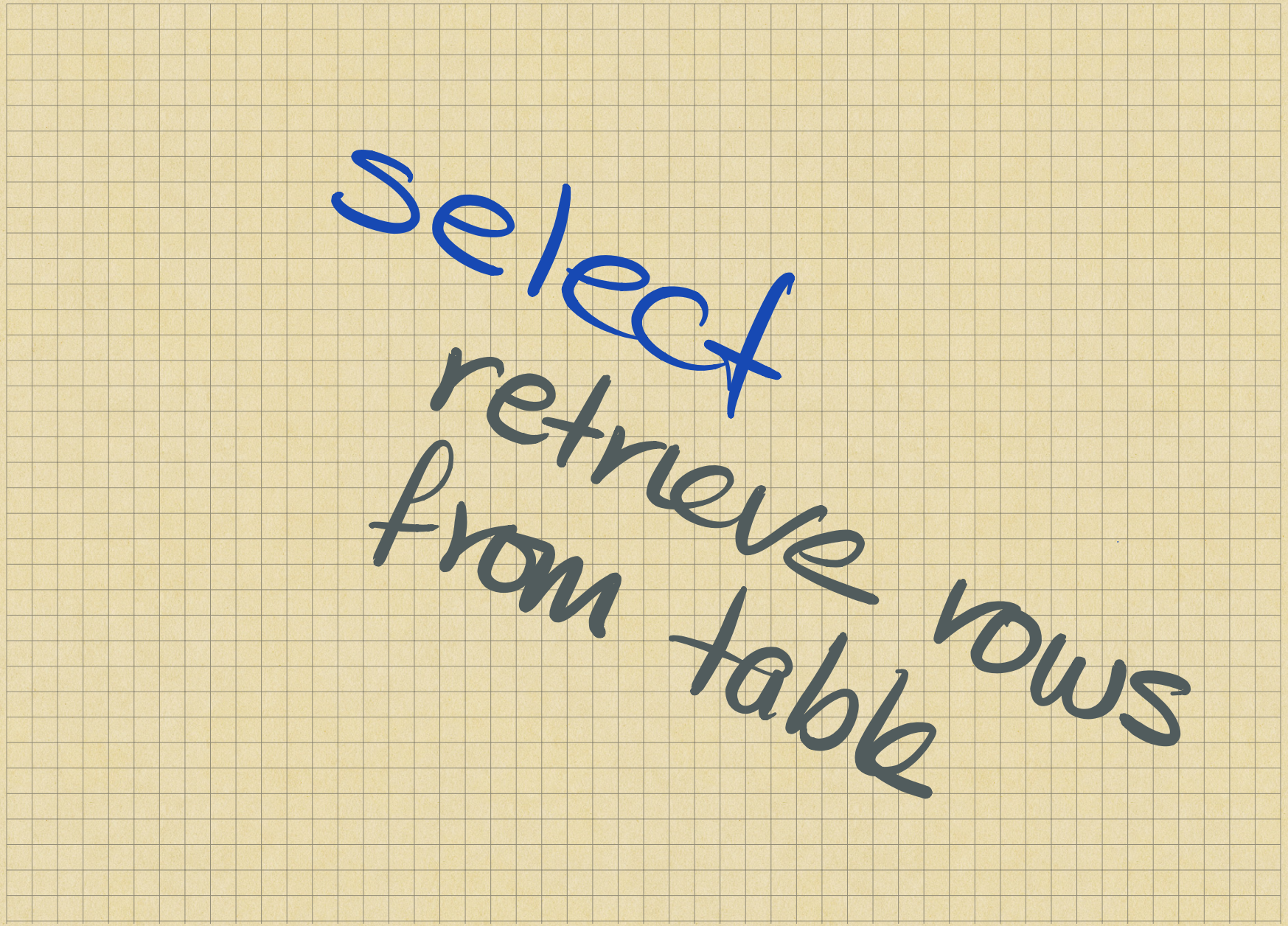
(vector of values)

- file already exists
- values are in order of field names
- insert a new row of values into the table: write these values into the file
  - insert each of these values along with the record number into corresponding index structure.



indices are multimaps of strings & longs:

{"Harris": 73}


select
retrieve rows
from table

set_fields:

Set the fields vector

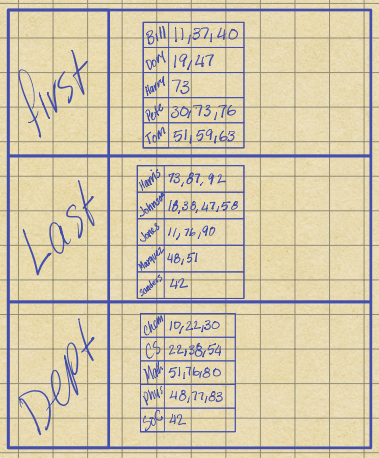
for every field:

push an index into the indices map

push the name into the names map

push the field name into _fields_names

indices:



_field_map

first	0
Last	1
Dep	2