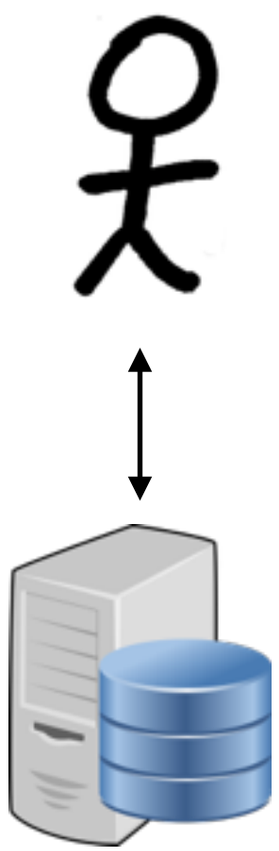
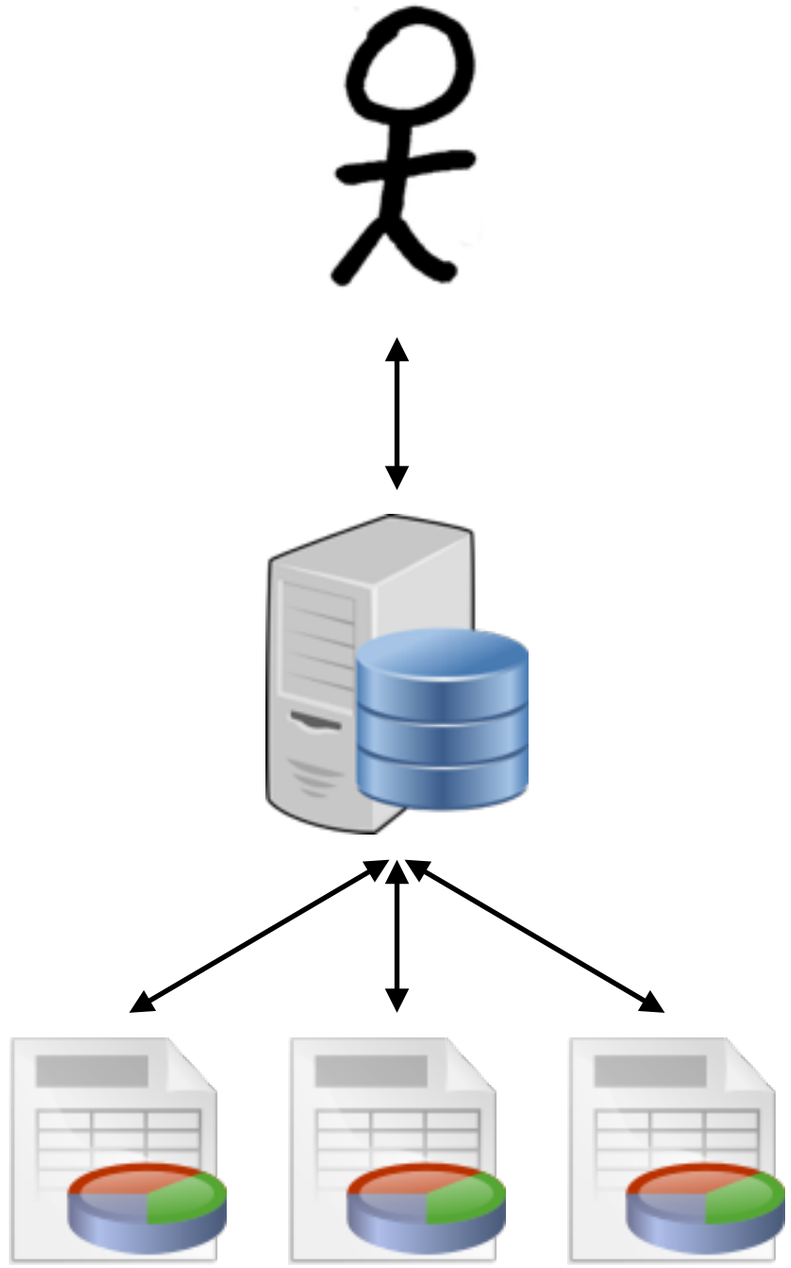
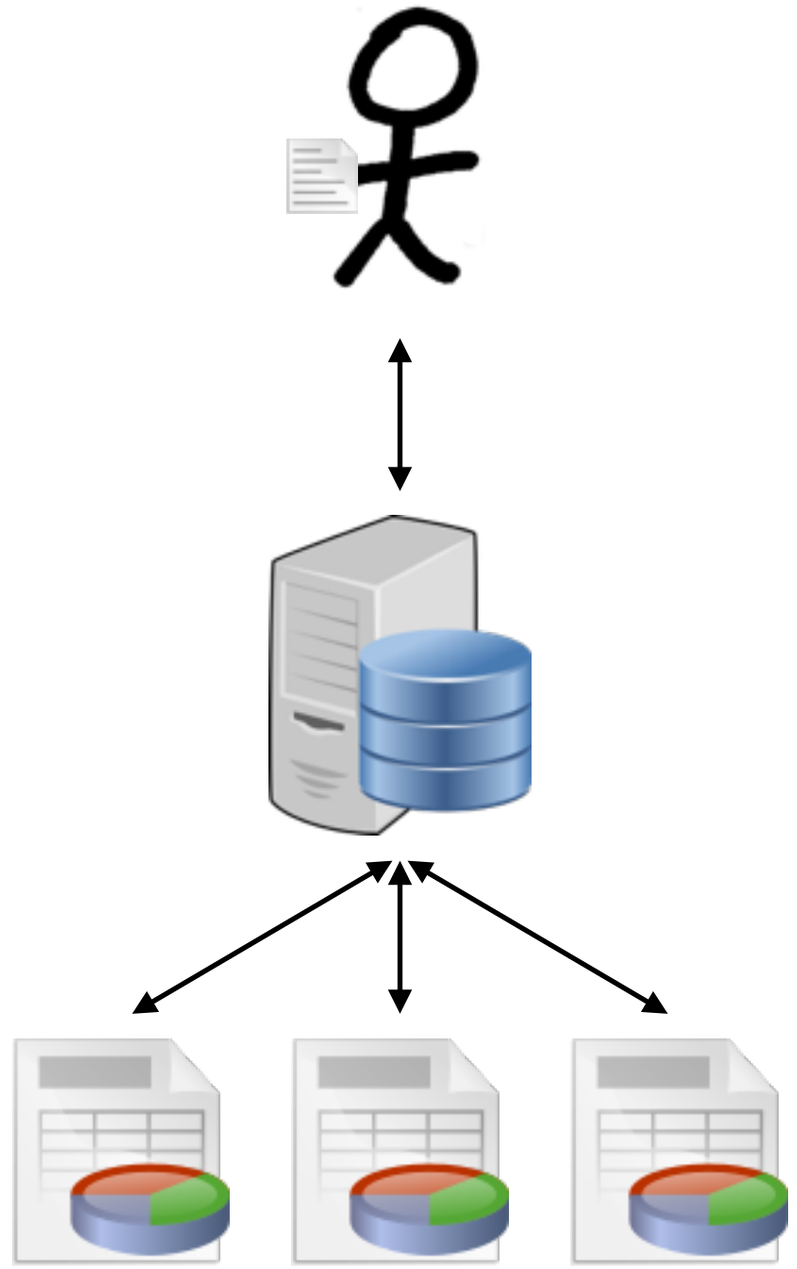
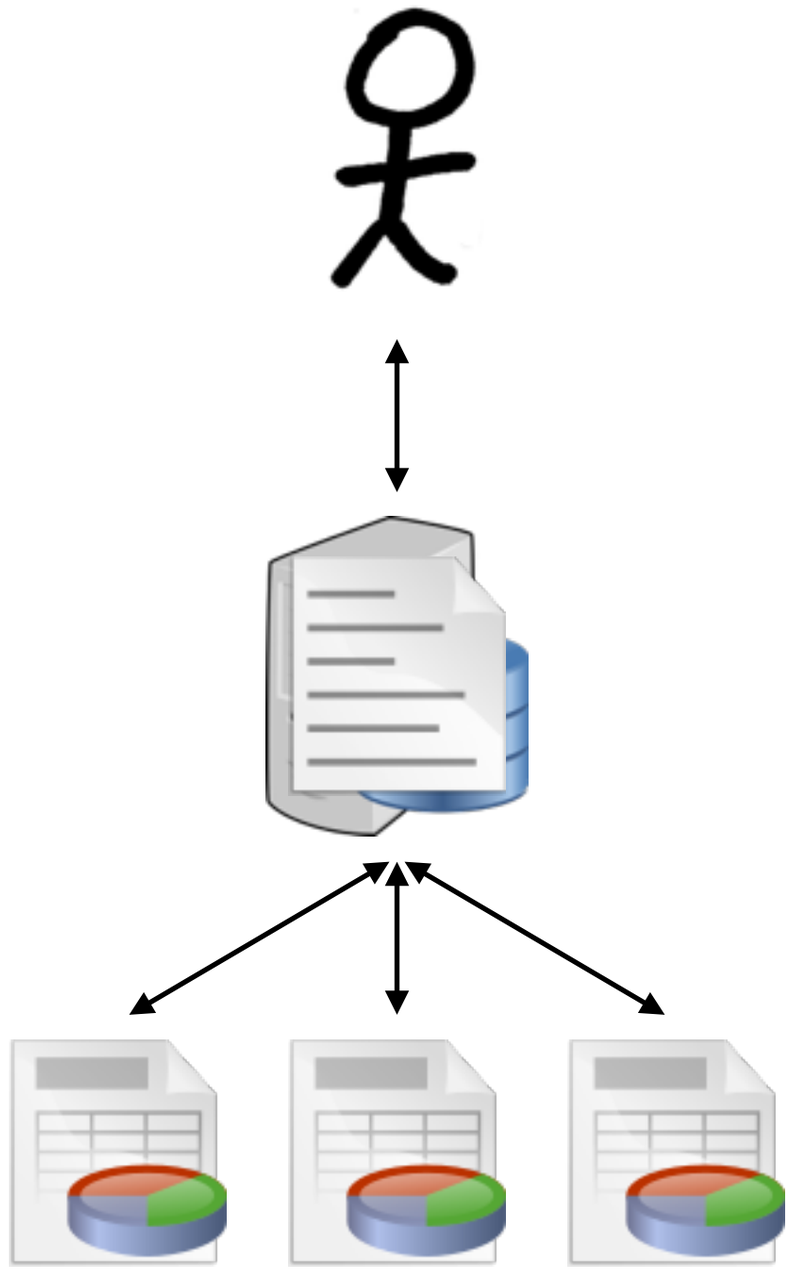
NoDB / RAW

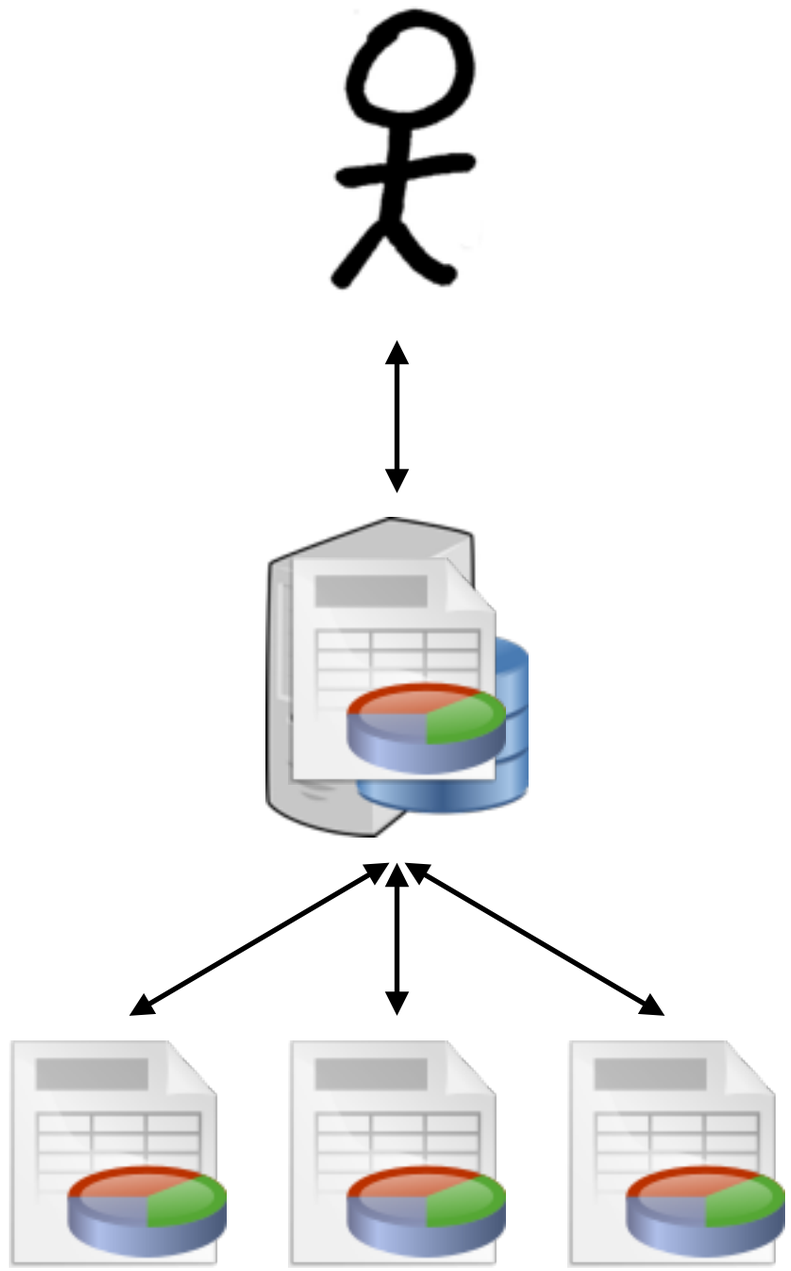
Oct 8, 2017

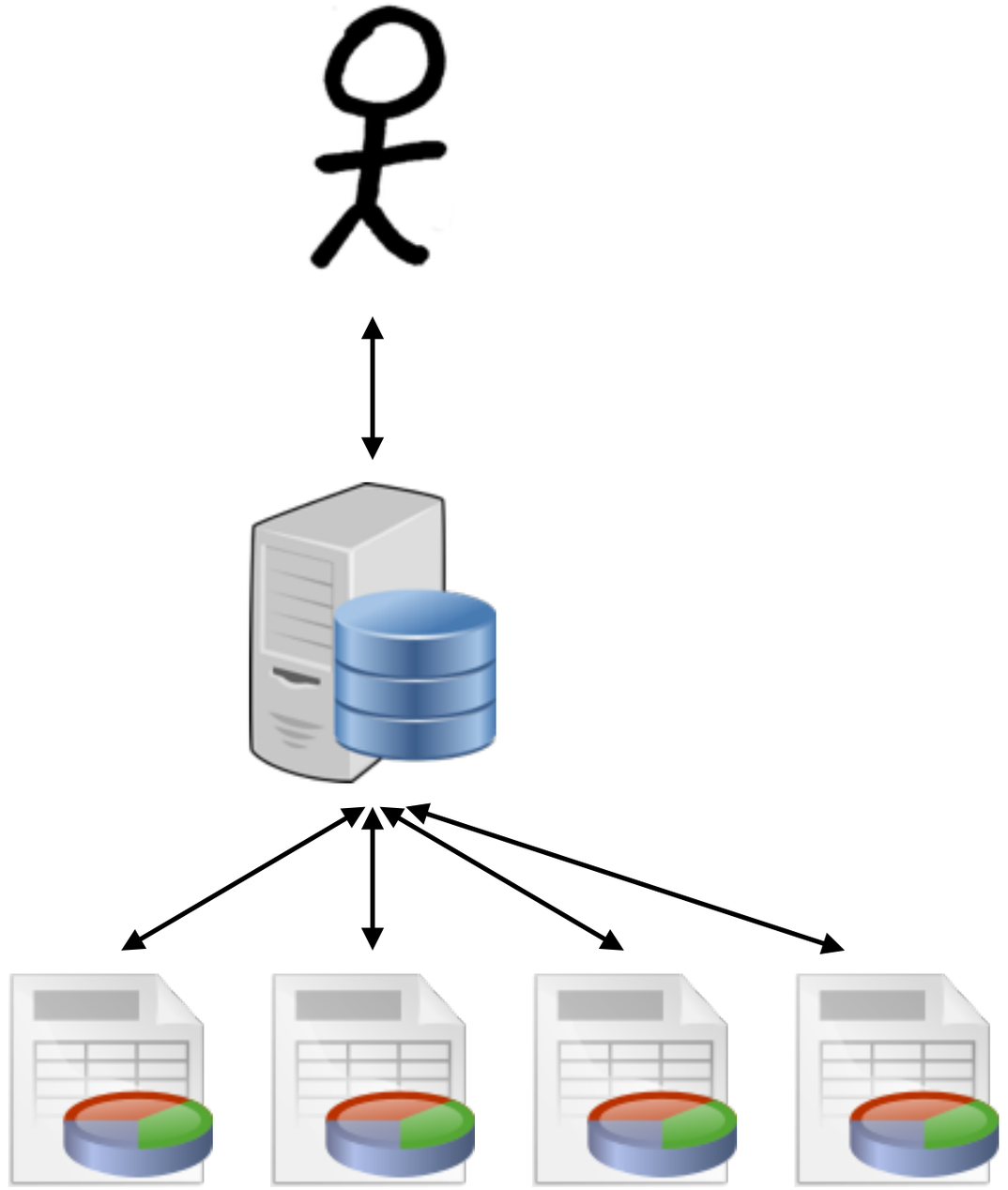


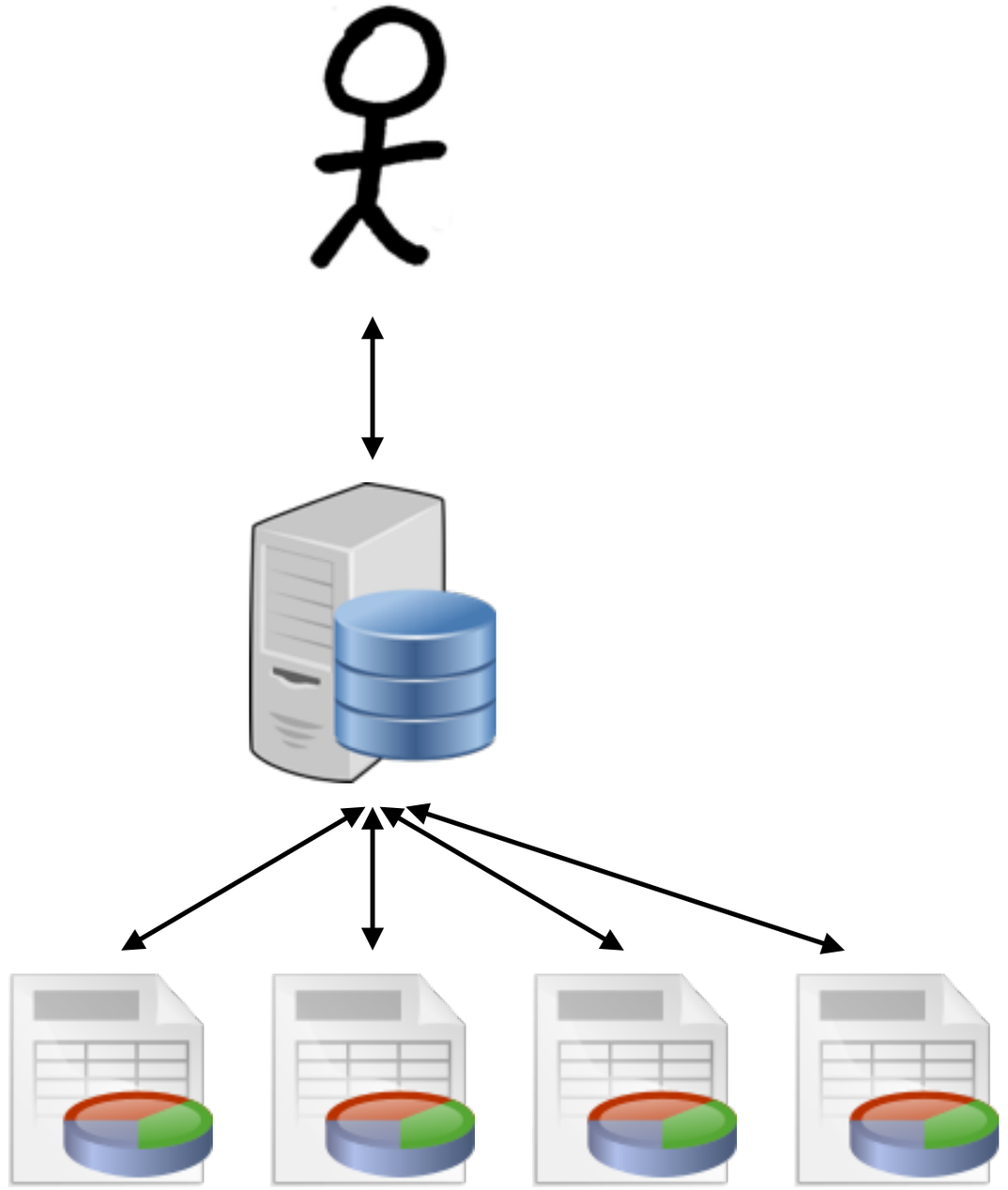


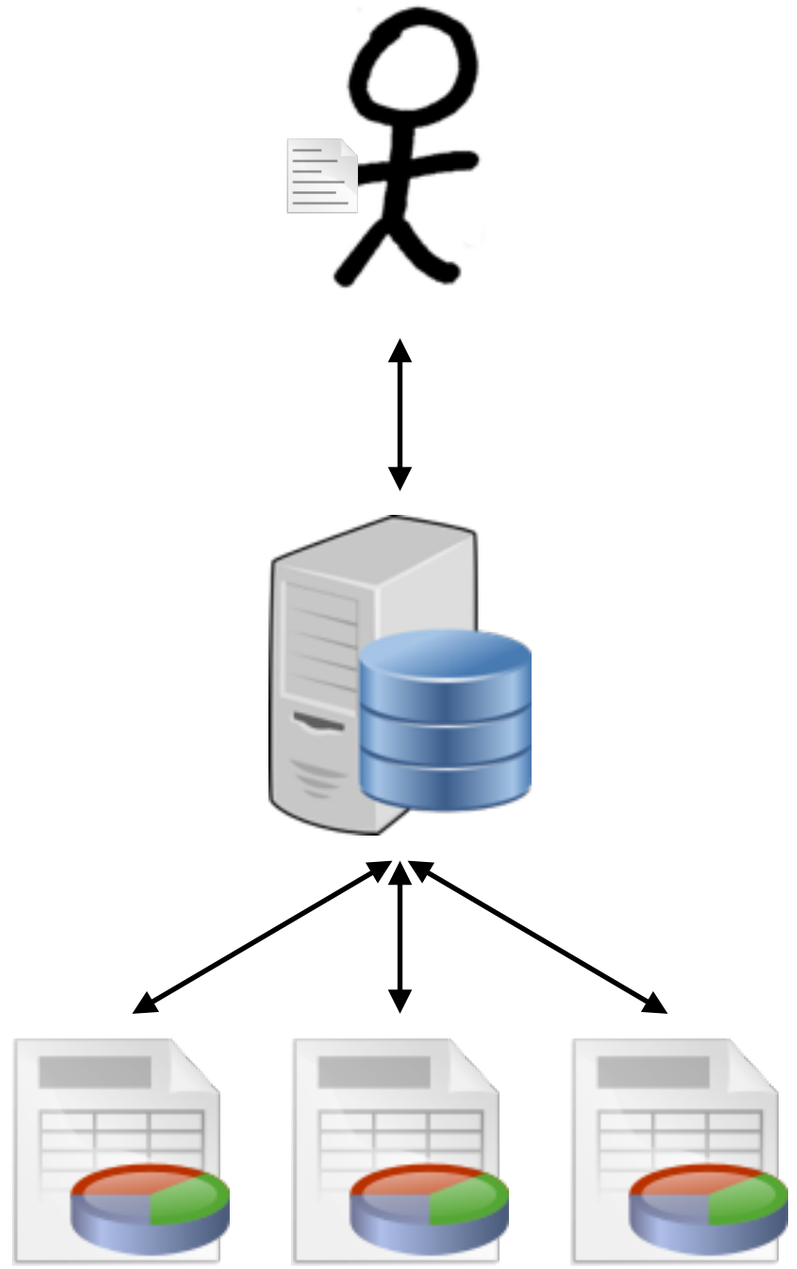


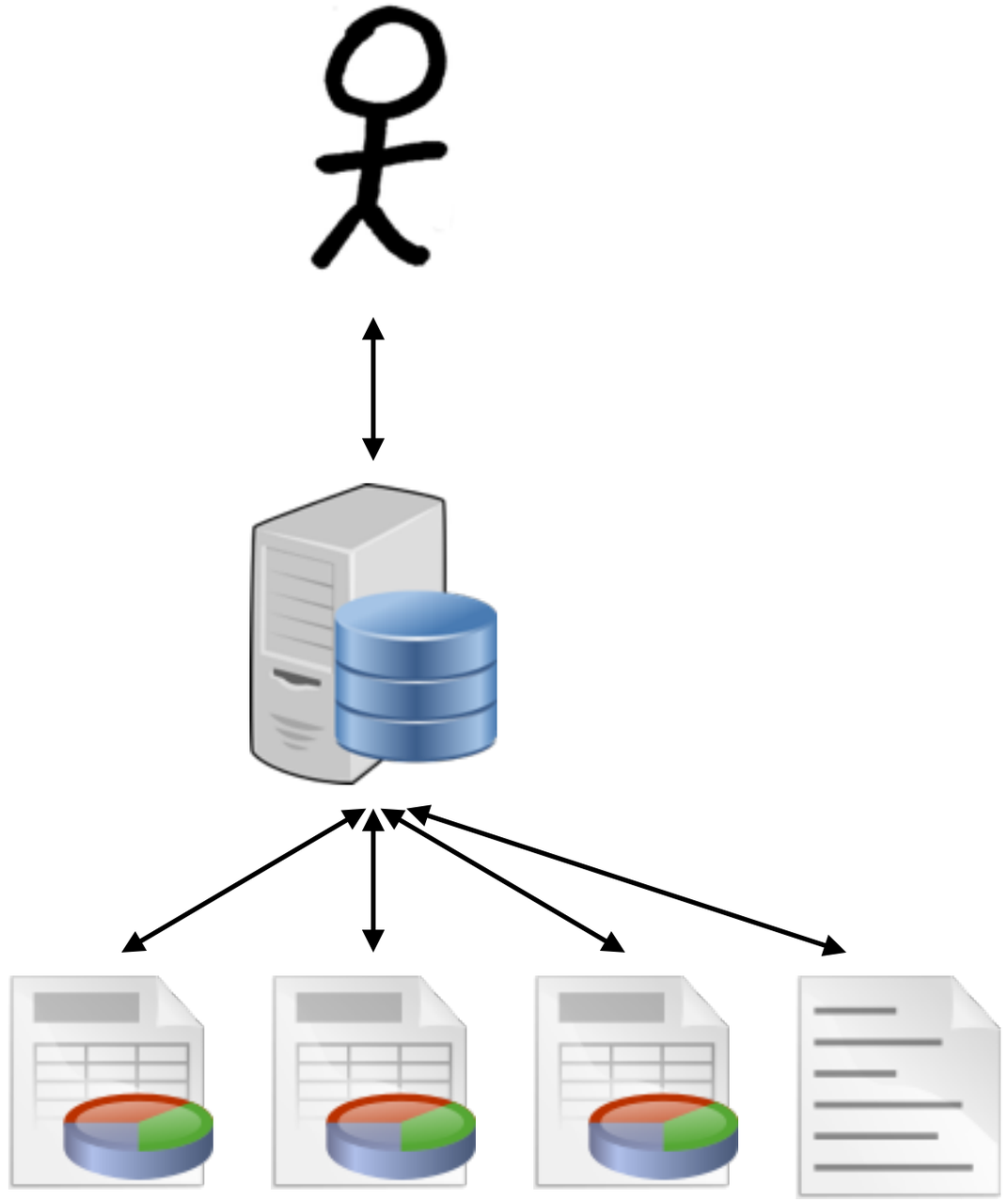












Why?

- Control over data.
  - The database needs the canonical version
- Load times
  - Copying data is expensive.

External Tables

- Supported by SQLite, Postgres, Oracle, DB2, …
- Read only access to raw data files
  - … but very slow

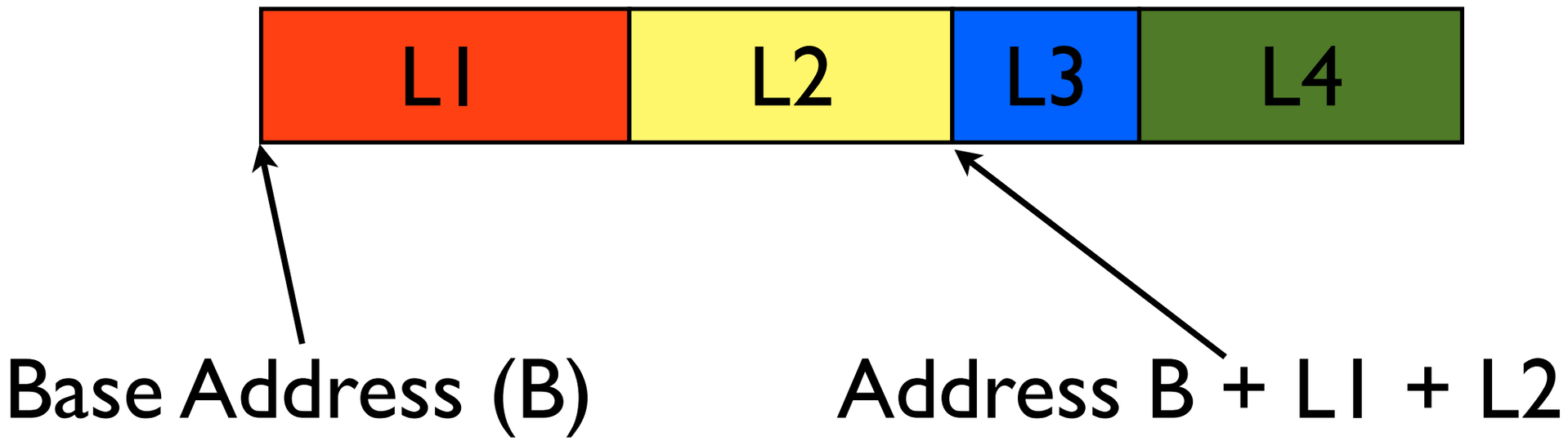
NoDB / RAW

- Parsing is slow
  - … so cut out unnecessary parsing steps.
  - … so cache intermediate parsing metadata.
  - … so just-in-time compile extraction code.
- Raw formats not optimized for database access.
  - … so cache parsed data in the database.

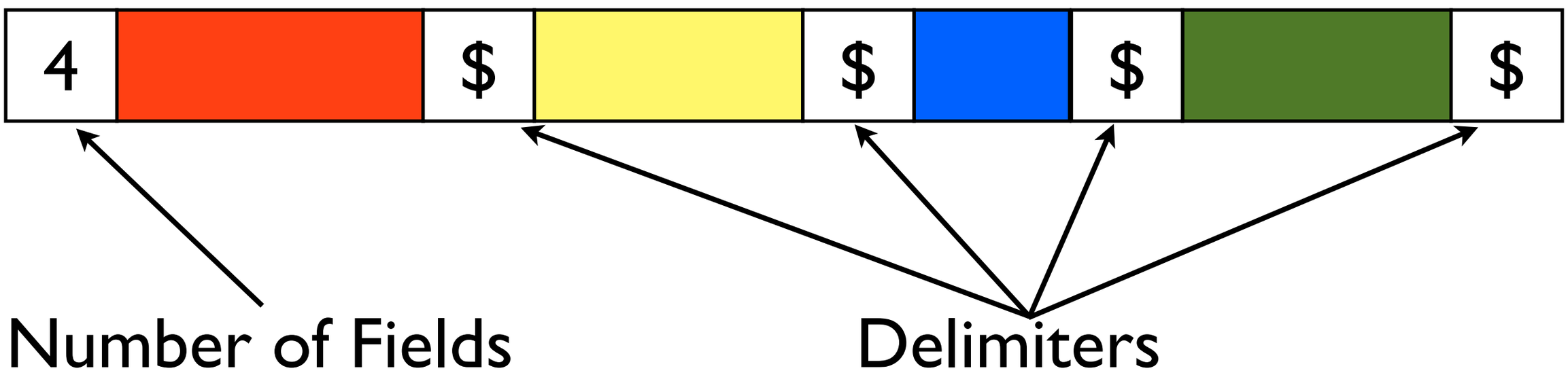
Data Organization

- How do we store data?
  - How are records represented on-disk? (Serialization)
    - How are records stored within a page?
    - How are pages organized in a file?
    - What other metadata do we need?
- Our solutions must also be persisted to disk.

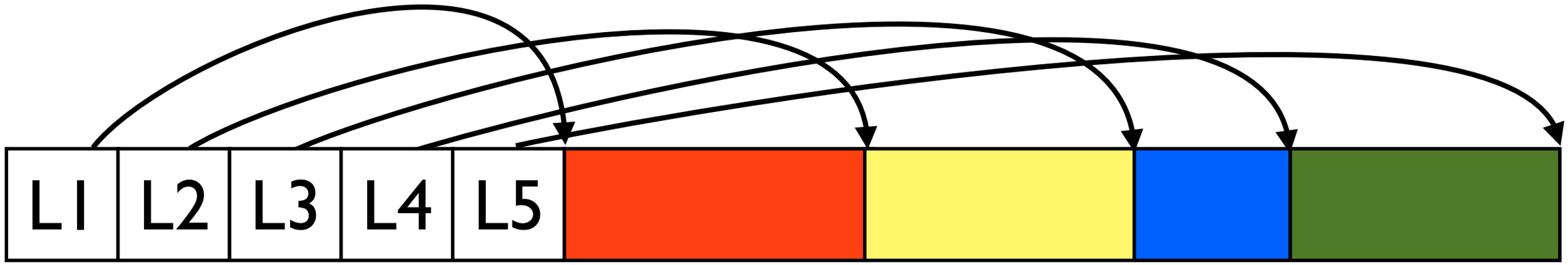
Record (Tuple) Formats



Record (Tuple) Formats



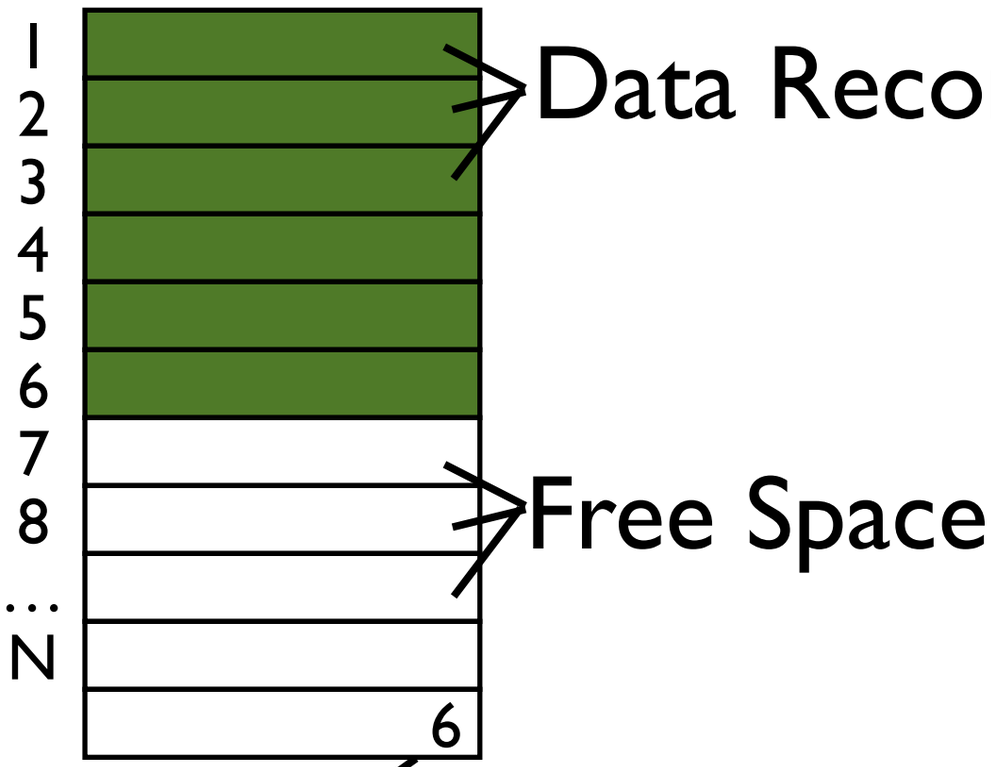
Record (Tuple) Formats



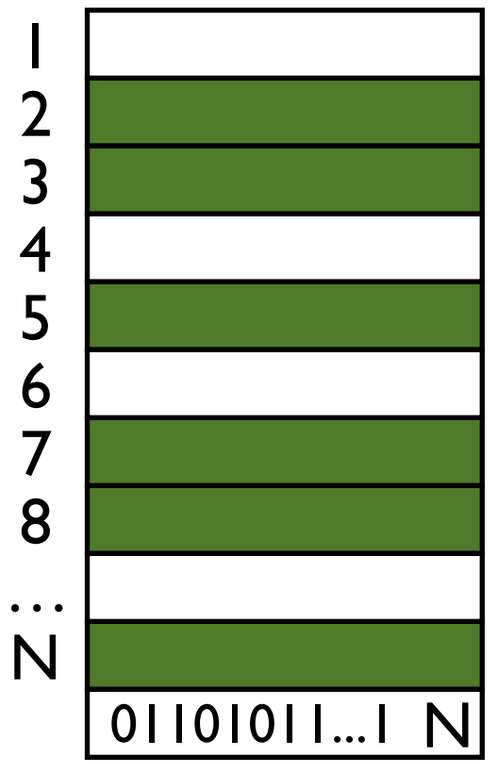
Array of Field Offsets

Page Formats

Packed



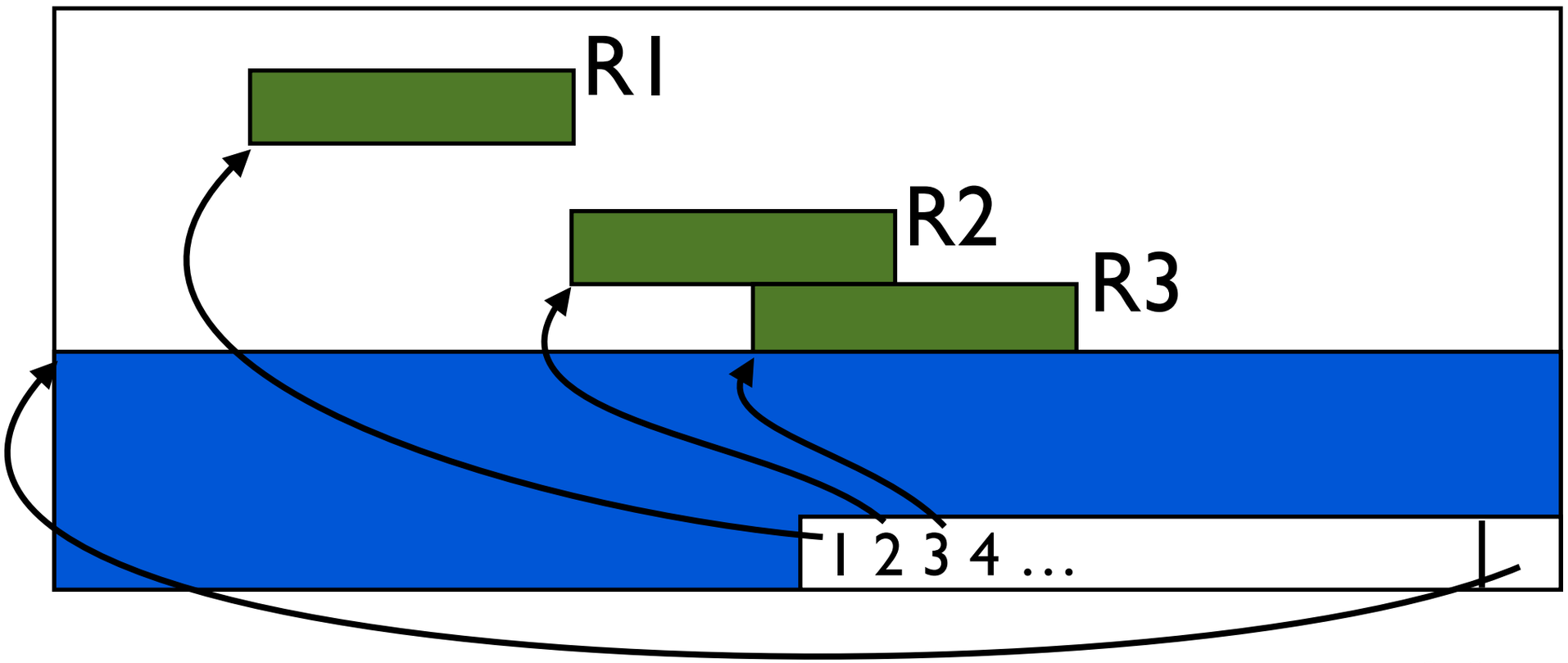
Unpacked, Bitmap



Bit array of occupied slots (and size of page)

Page Formats

Variable Size Records



Anatomy of a CSV file

Year	Make	Model	Description	Price
1997	Ford	E350	ac, abs, moon	3000.00
1999	Chevy	Venture "Extended Edition"		4900.00
1999	Chevy	Venture "Extended Edition, Very Large"		5000.00
1996	Jeep	Grand Cherokee	MUST SELL! air, moon roof, loaded	4799.00

```
Year,Make,Model,Description,Price
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,"Venture ""Extended Edition""","",4900.00
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00
1996,Jeep,Grand Cherokee,"MUST SELL!
air, moon roof, loaded",4799.00
```

The Filesystem View

Year,Make,Model,Description,Price\n1997,Ford,E350,"ac, abs, moon”,
3000.00\n1999,Chevy,"Venture ""Extended Edition”""","",4900.00\n1999,Chevy,"Venture ""Extended
Edition, Very Large”"",,5000.00\n1996,Jeep,Grand Cherokee,"MUST SELL!\nair, moon roof, loaded",
4799.00

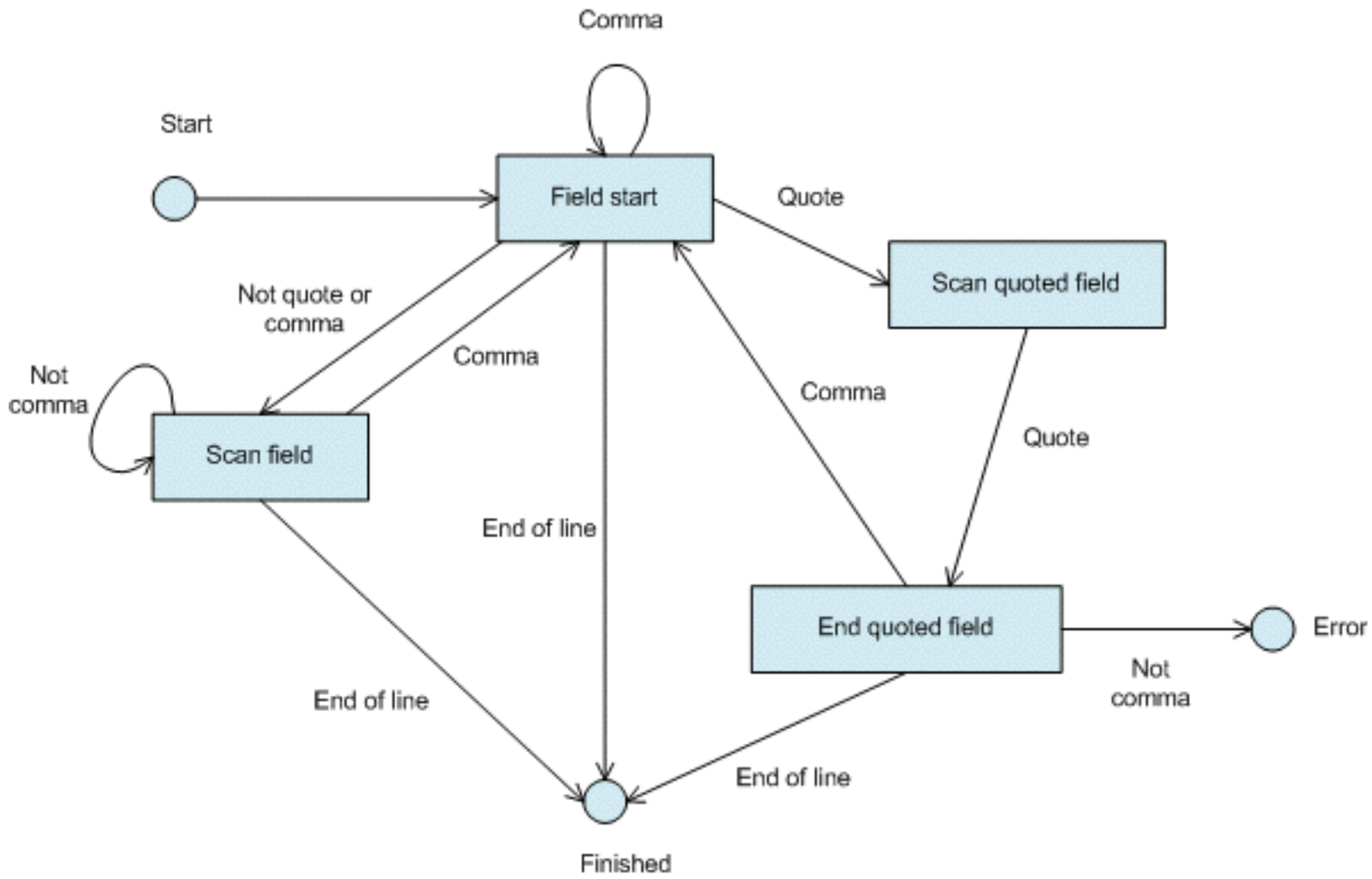
The Filesystem View

Year,Make,Model,Description,Price**\n**1997,Ford,E350,"ac, abs, moon”,
3000.00**\n**1999,Chevy,"Venture ""Extended Edition”""","",4900.00**\n**1999,Chevy,"Venture ""Extended Edition, Very Large”""",,5000.00**\n**1996,Jeep,Grand Cherokee,"MUST SELL!**\n**air, moon roof, loaded",
4799.00

The Filesystem View

Year,Make,Model,Description,Price**\n**1997,Ford,E350,"ac, abs, moon”,
3000.00**\n**1999,Chevy,"Venture ""Extended Edition”””,"",4900.00**\n**1999,Chevy,"Venture ""Extended Edition, Very Large”””,,5000.00**\n**1996,Jeep,Grand Cherokee,"MUST SELL!**\n**air, moon roof, loaded",
4799.00

Field Parsing



https://sourcemaking.com/design_patterns/state/delphi

Field Parsing

1997,Ford,E350,"ac, abs, moon",3000.00

1999,Chevy,"Venture ""Extended Edition""","",4900.00

1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00

1996,Jeep,Grand Cherokee,"MUST SELL!\nair, moon roof, loaded",4799.00

Field Parsing

1997,Ford,E350,"ac, abs, moon",3000.00

1999,Chevy,"Venture ""Extended Edition""","",4900.00

1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00

1996,Jeep,Grand Cherokee,"MUST SELL!\nair, moon roof, loaded",4799.00

Field Parsing

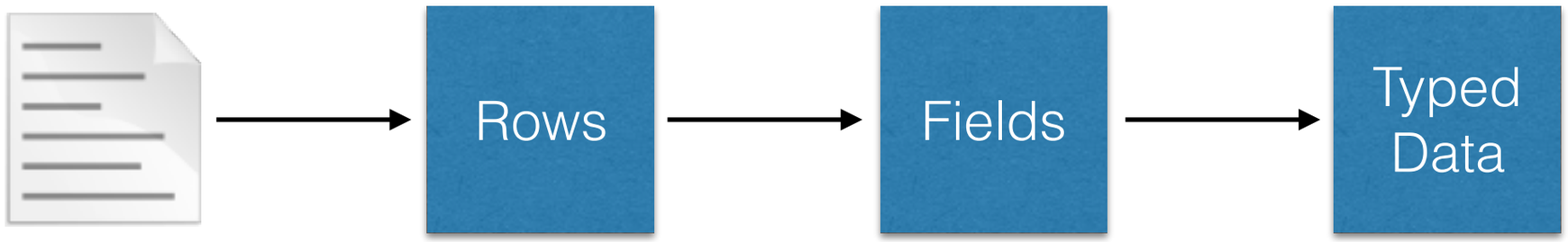
1997,Ford,E350,"ac, abs, moon",3000.00

1999,Chevy,"Venture ""Extended Edition""","",4900.00

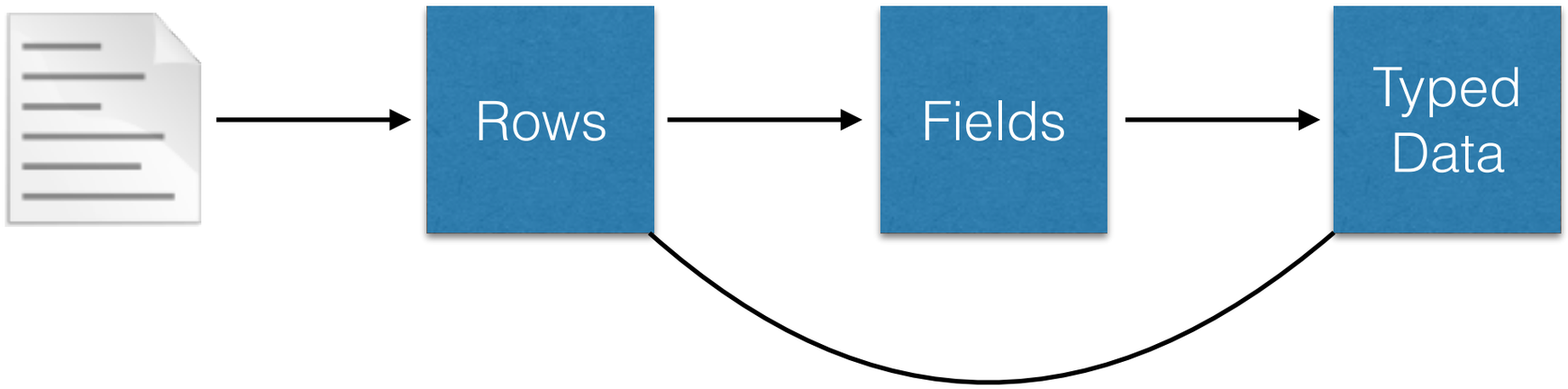
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00

1996,Jeep,Grand Cherokee,"MUST SELL!\nair, moon roof, loaded",4799.00

Overview

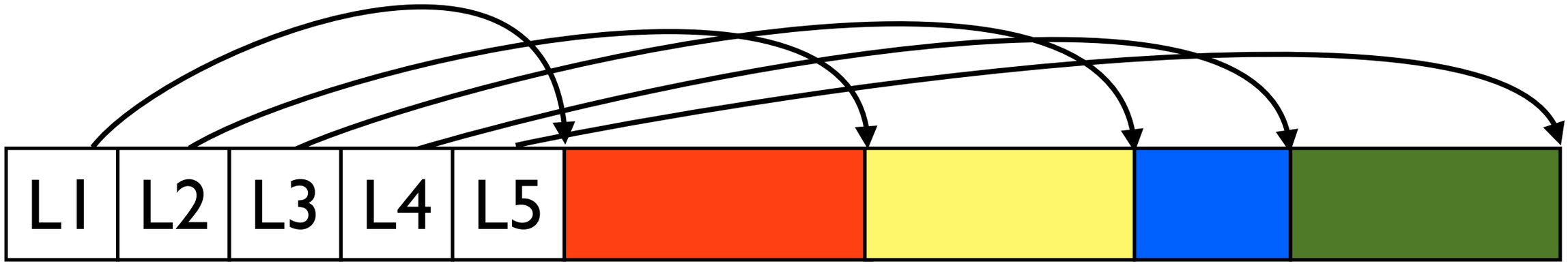


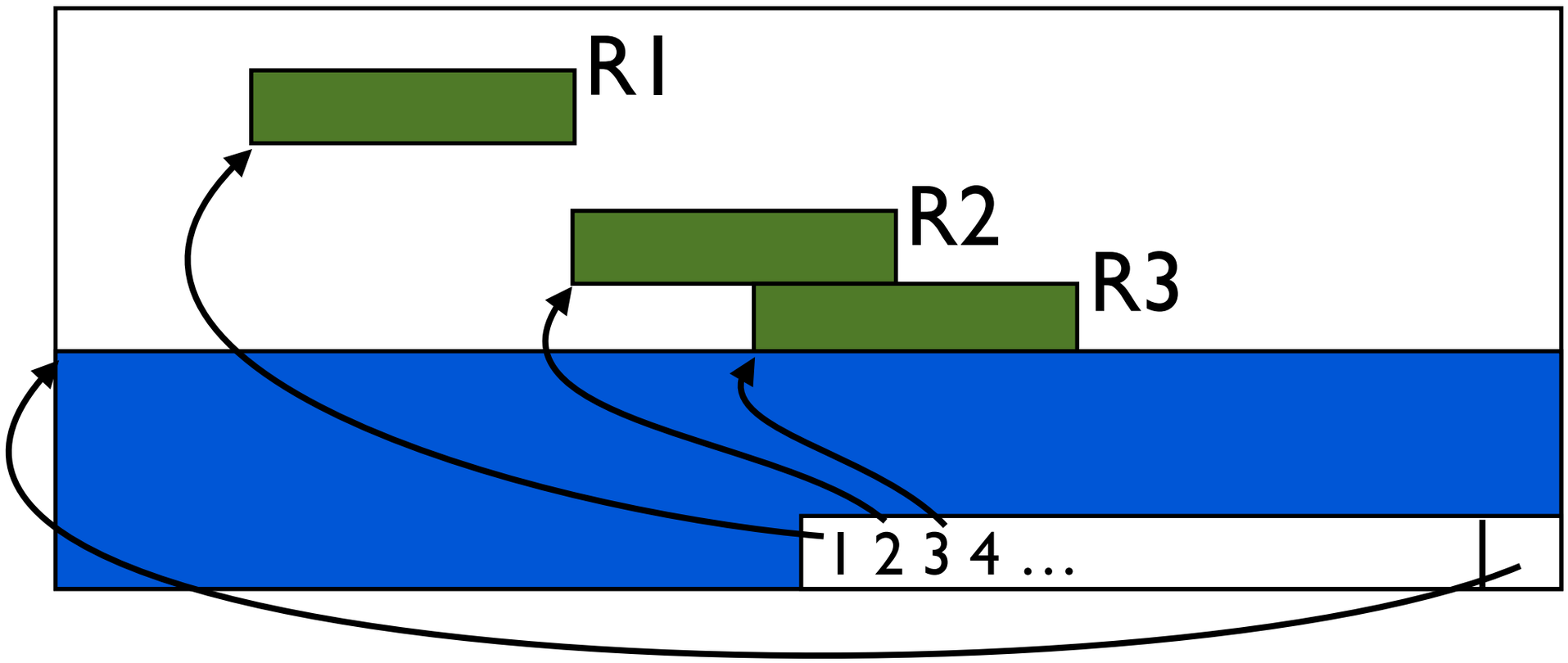
Overview



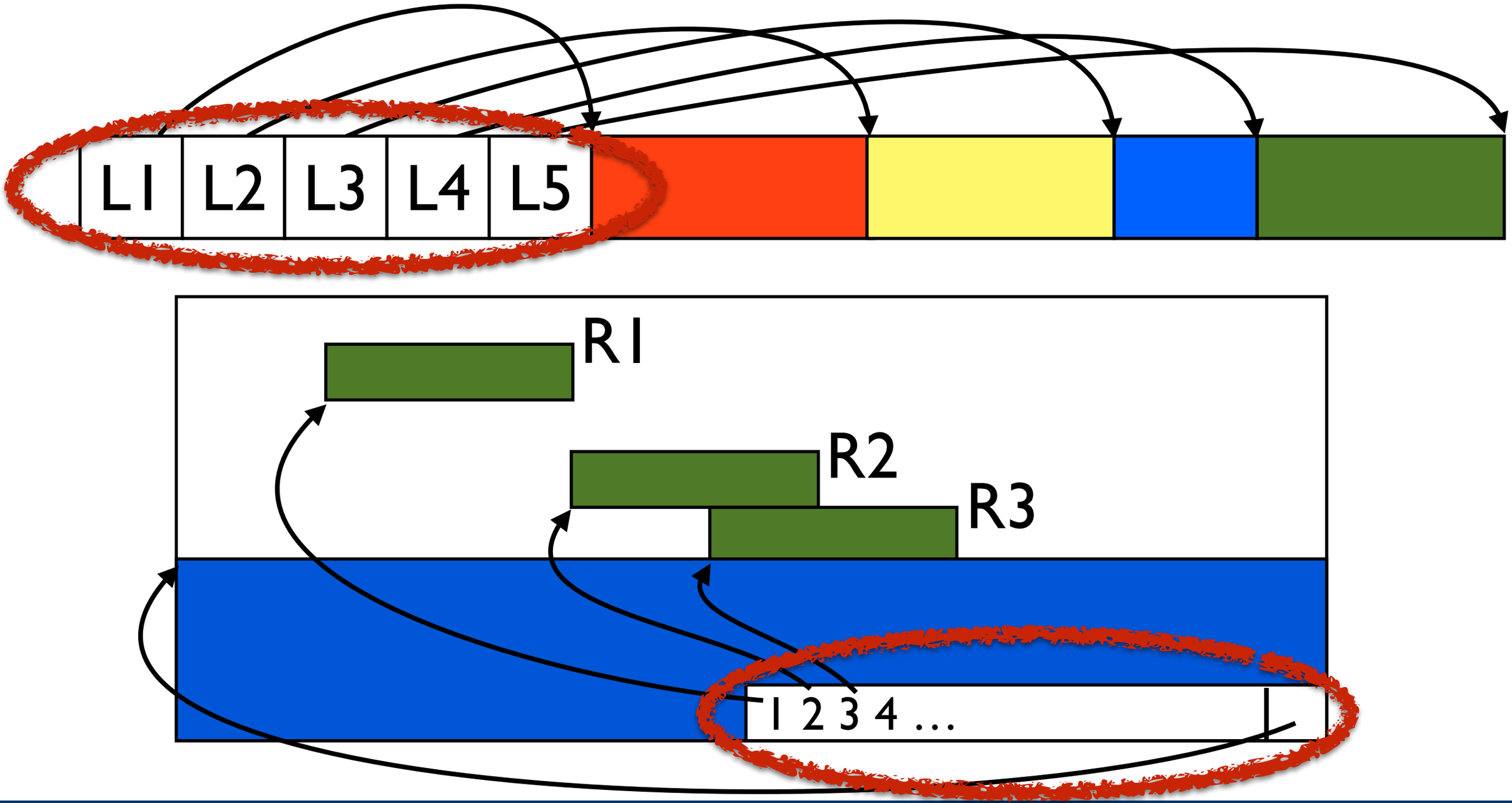
Core Idea: Do this part once only

Avoiding Re-Splitting





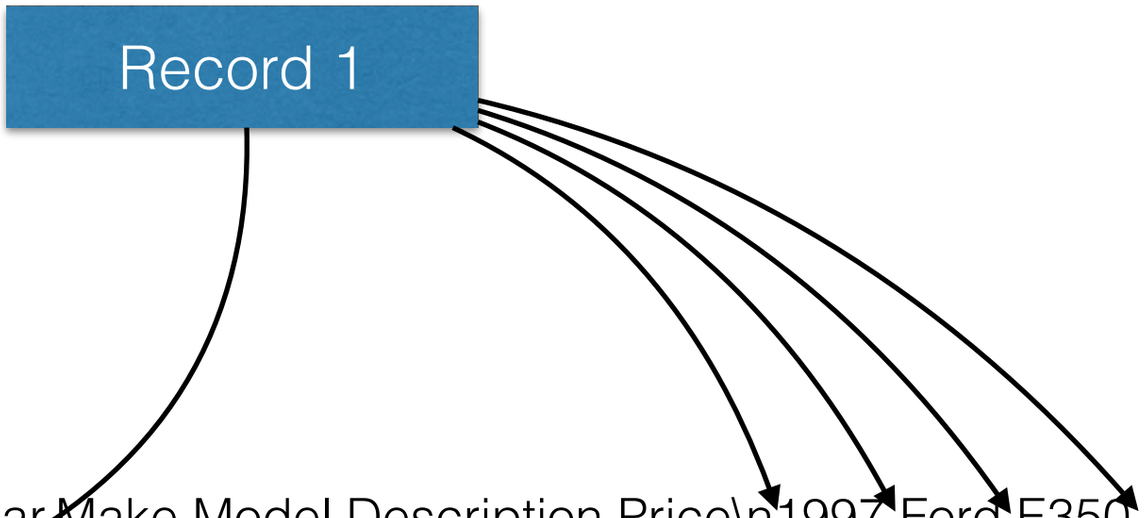
Avoiding Re-Splitting



Position Map

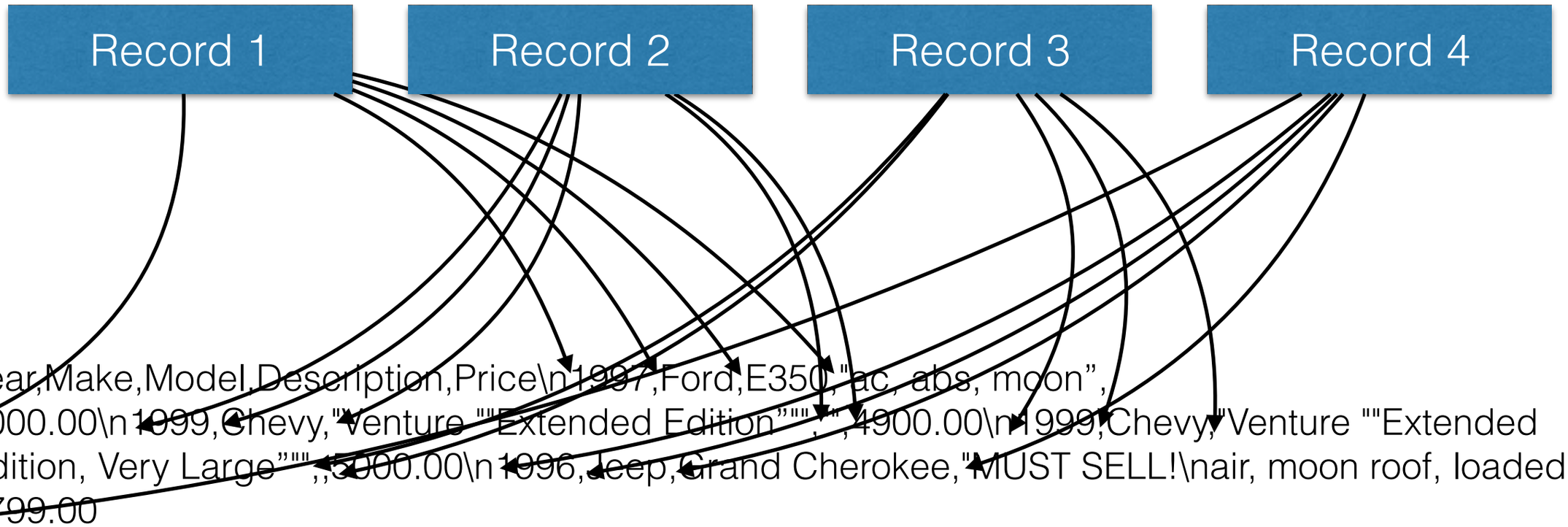
Idea 1: Index the data

Position Map

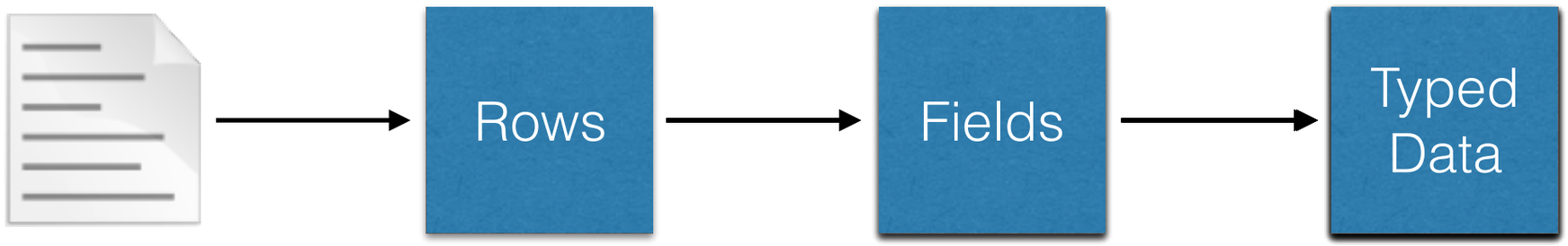


Year,Make,Model,Description,Price\n1997,Ford,E350,"ac, abs, moon”,
3000.00\n1999,Chevy,"Venture ""Extended Edition”"",",4900.00\n1999,Chevy,"Venture ""Extended
Edition, Very Large”"",,5000.00\n1996,Jeep,Grand Cherokee,"MUST SELL!\nair, moon roof, loaded",
4799.00

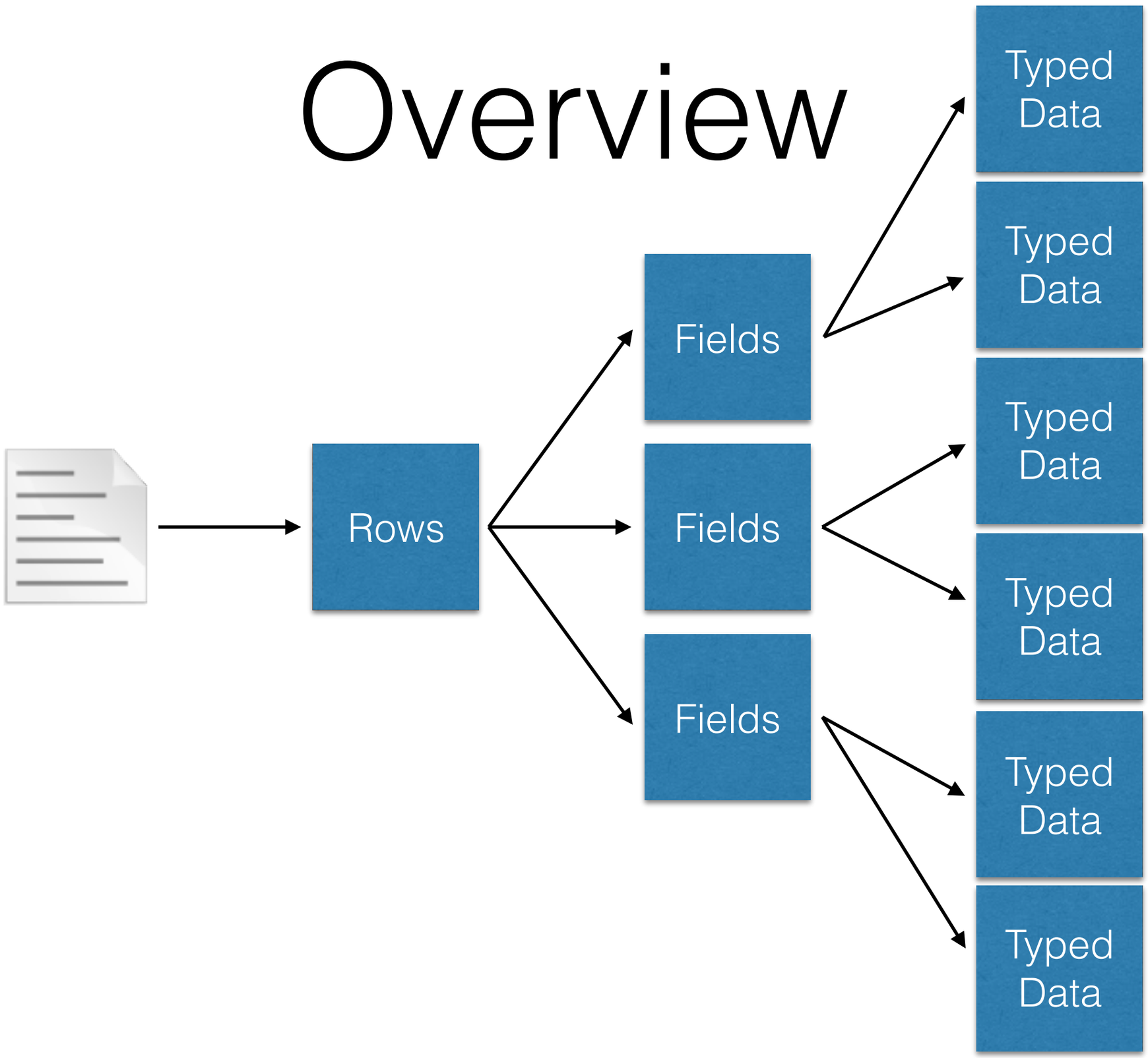
Position Map



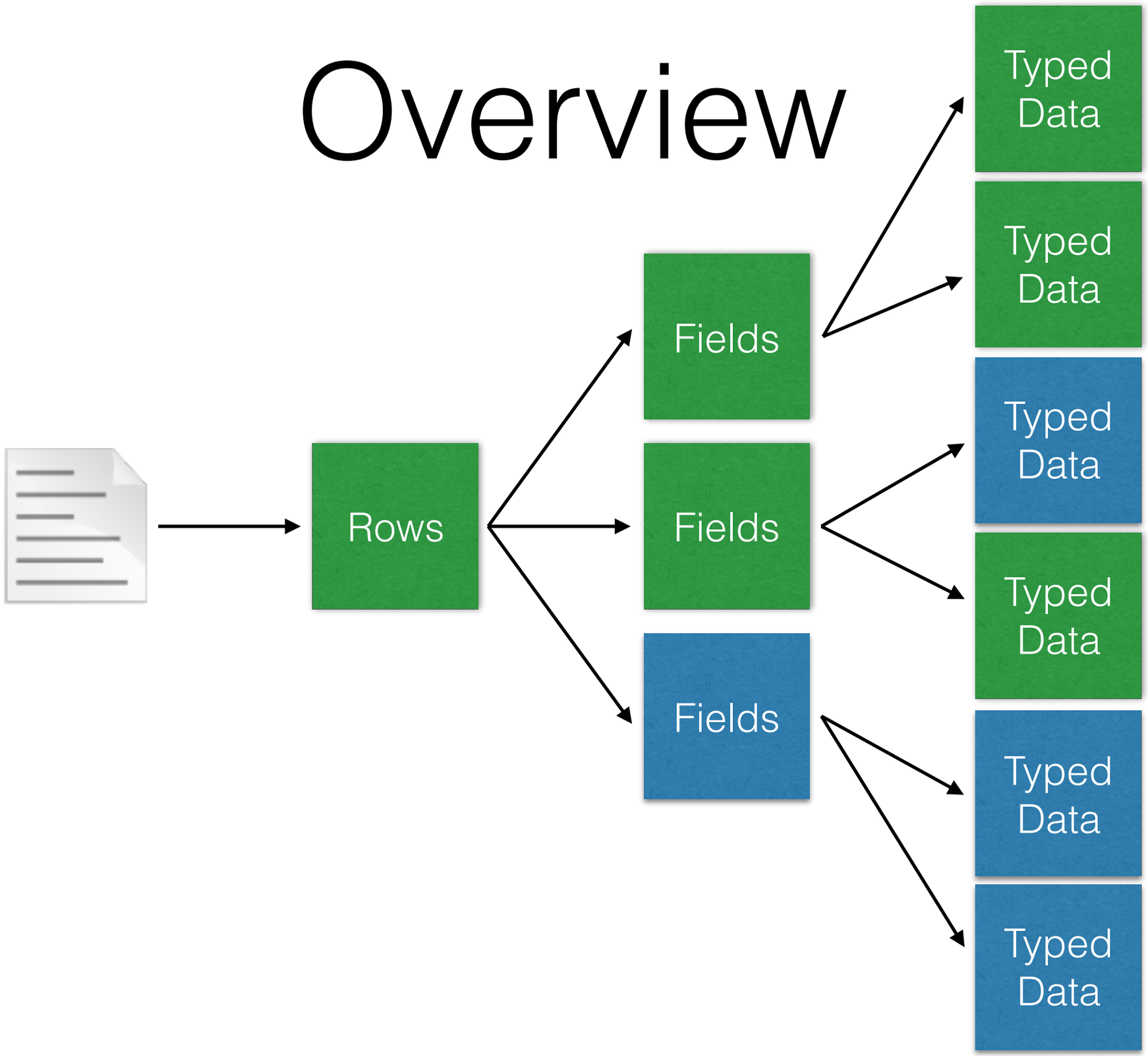
Overview



Overview


Rows
Fields
Fields
Fields
Typed Data
Typed Data
Typed Data
Typed Data
Typed Data
Typed Data

Overview


Rows
Fields
Fields
Fields
Typed Data
Typed Data
Typed Data
Typed Data
Typed Data
Typed Data

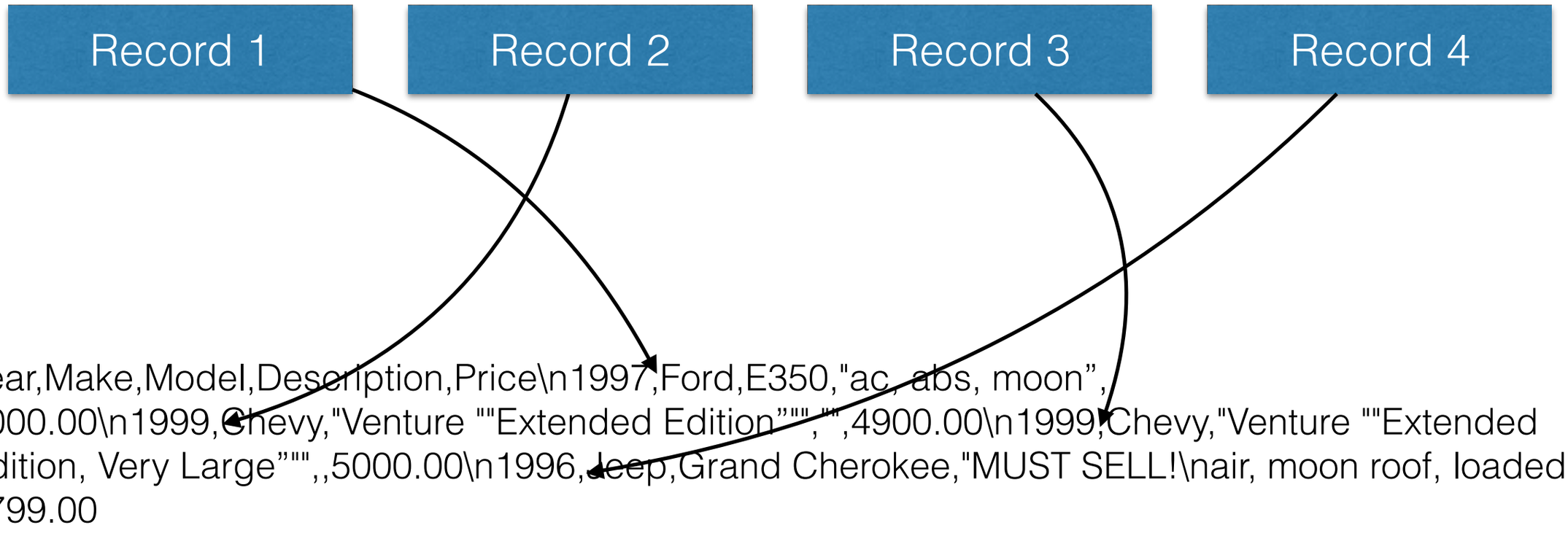
Position Maps

Idea 2: Index Lazily

Position Map

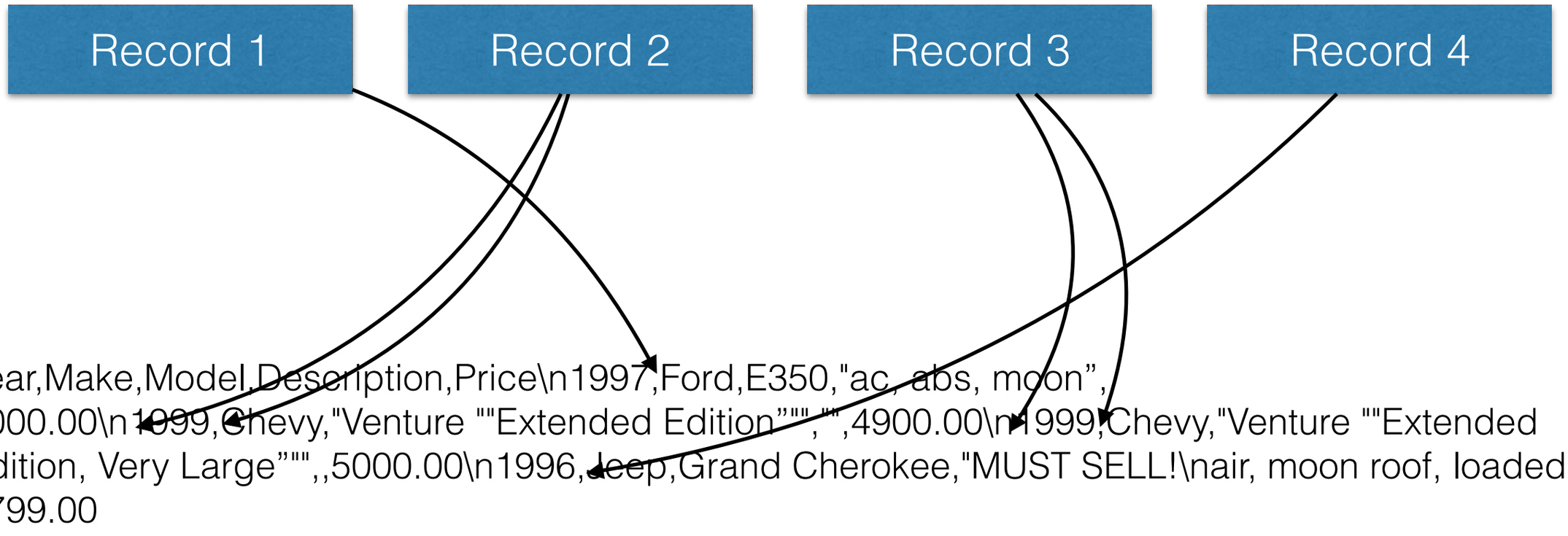
Year,Make,Model,Description,Price\n1997,Ford,E350,"ac, abs, moon”,
3000.00\n1999,Chevy,"Venture ""Extended Edition”””,"",4900.00\n1999,Chevy,"Venture ""Extended
Edition, Very Large”””,,5000.00\n1996,Jeep,Grand Cherokee,"MUST SELL!\nair, moon roof, loaded",
4799.00

Position Map




```
SELECT Make FROM Cars
```

Position Map

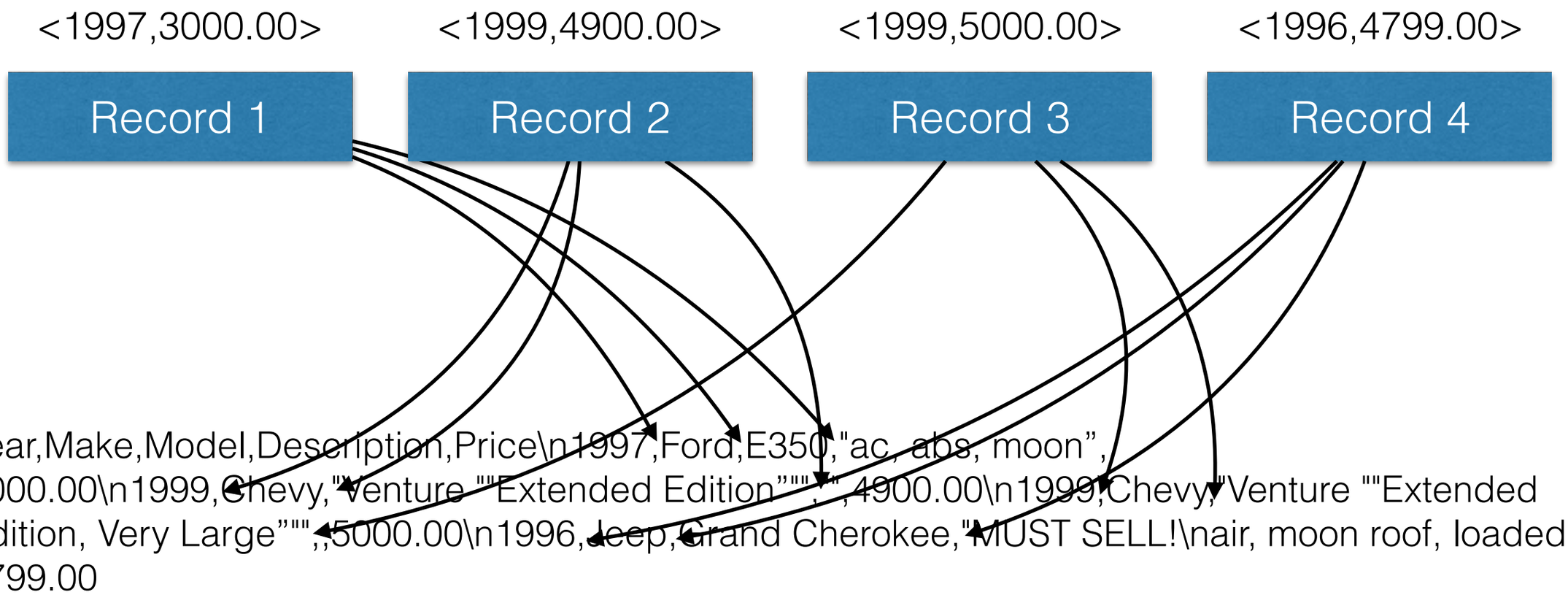



```
SELECT Make FROM Cars
SELECT Year FROM Cars WHERE Make = ‘Chevy’
```

Position Maps

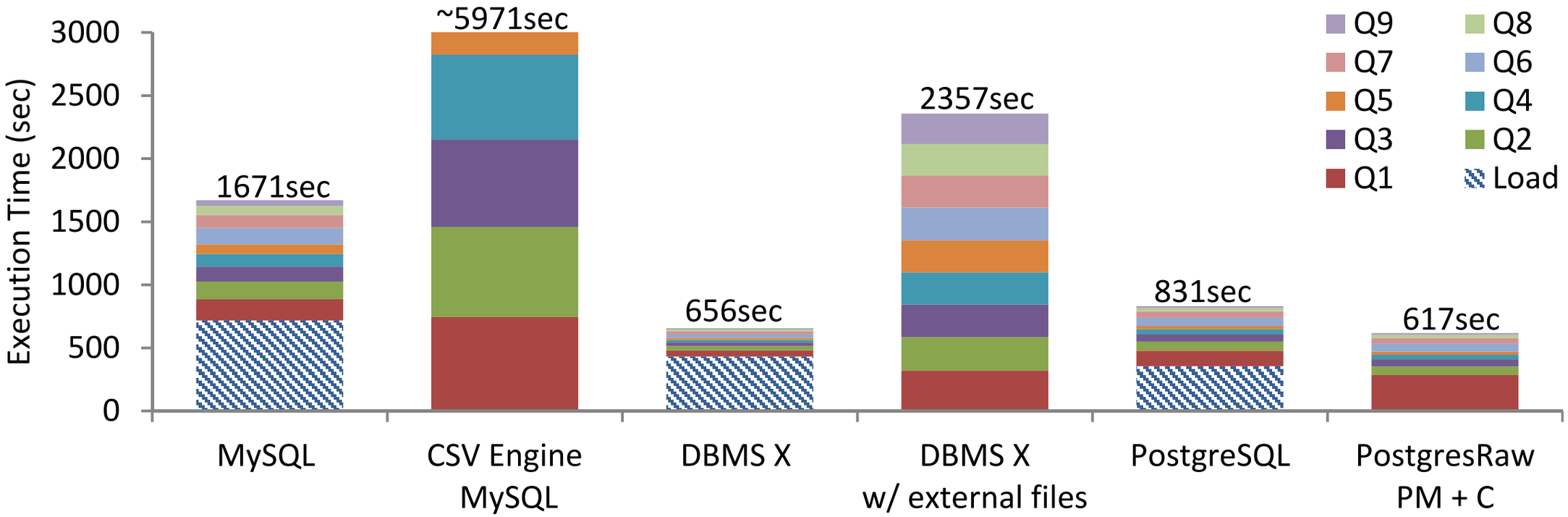
Idea 3: Cache “small” parsed values

Position Map



Year,Make,Model,Description,Price\n1997,Ford,E350,"ac, abs, moon”,3000.00\n1999,Chevy,"Venture ""Extended Edition”"",",4900.00\n1999,Chevy,"Venture ""Extended Edition, Very Large”"",,5000.00\n1996,Jeep,Grand Cherokee,"MUST SELL!\nair, moon roof, loaded",4799.00

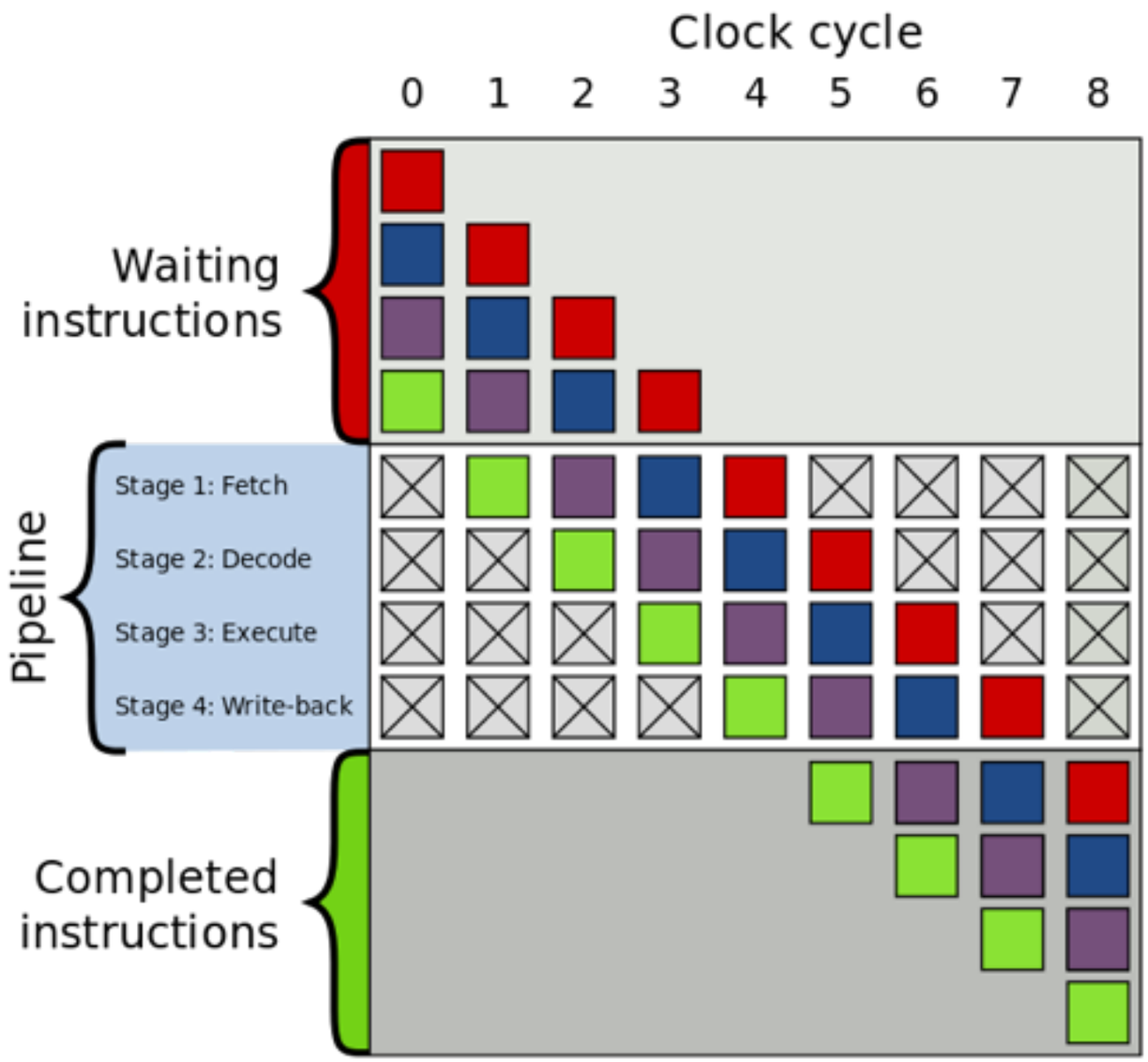
Performance



JIT-ed Parsers

Idea 4: Eliminate Branching

Branch Prediction



https://en.wikipedia.org/wiki/File:Pipeline,_4_stage.svg

JIT-ed Parsers

```
for every column {
   char *raw // raw data
   Datum *datum // loaded data

   // read field from file
   raw = readNextFieldFromFile(file)

   switch (schemaDataType[column]) {
      case IntType: datum = convertToInteger(raw)
      case FloatType: datum = convertToFloat(raw)
...
}
```

JIT-ed Parsers

```
for every column {
    char *raw // raw data
    Datum *datum // loaded data

    // read field from file
    raw = readNextFieldFromFile(file)

    switch (schemaDataType[column]) {
        case IntType: datum = convertToInteger(raw)
        case FloatType: datum = convertToFloat(raw)
...
}
```

JIT-ed Parsers

```
for every column {
    char *raw // raw data
    Datum *datum // loaded data

    // read field from file
    raw = readNextFieldFromFile(file)

    switch (schemaDataType[column]) {
        case IntType: datum = convertToInteger(raw)
        case FloatType: datum = convertToFloat(raw)
...
}
```

JIT-ed Parsers

```
for every column {
    char *raw // raw data
    Datum *datum // loaded data

    // read field from file
    raw = readNextFieldFromFile(file)

    switch (schemaDataType[column]) {
        case IntType: datum = convertToInteger(raw)
        case FloatType: datum = convertToFloat(raw)
...
}
```

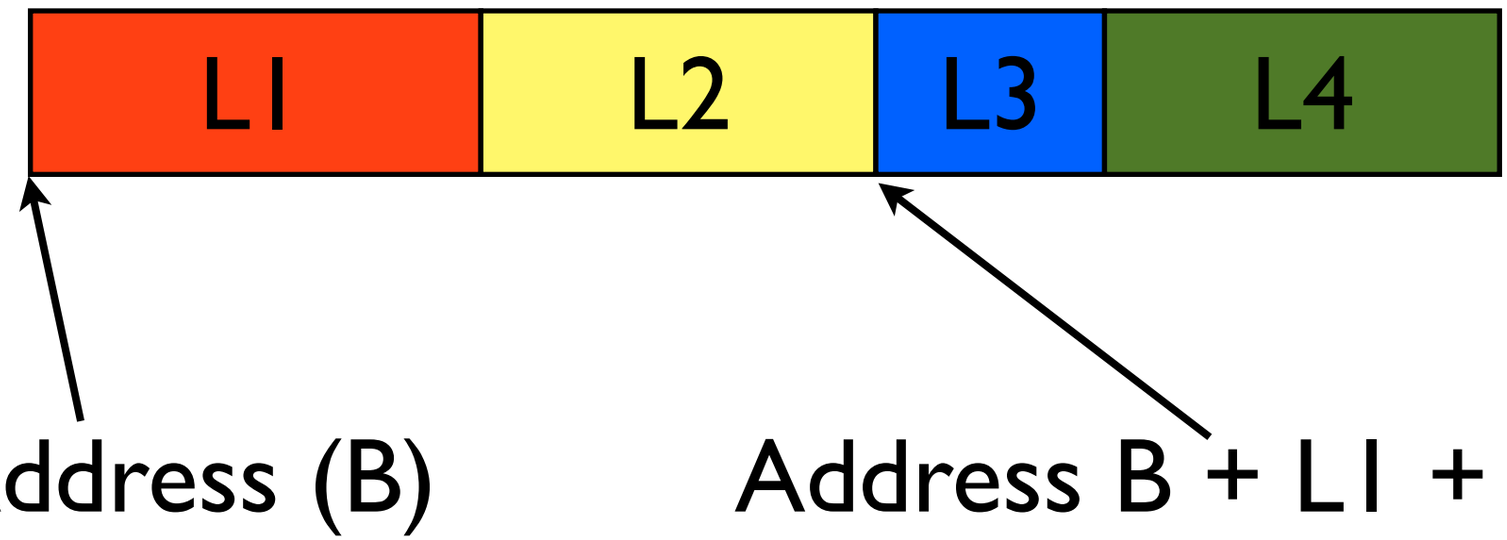
JIT-ed Parsers

```
Datum *datum[] // loaded data

datum[0] = convertToInteger(readNextFieldFromFile(file));
datum[1] = convertToString(readNextFieldFromFile(file));
datum[2] = convertToString(readNextFieldFromFile(file));
datum[3] = convertToString(readNextFieldFromFile(file));
datum[4] = convertToFloat(readNextFieldFromFile(file));
```

No Loop, No Ifs, No branching

More opportunities



JIT-ed Parsers

```
char *raw // raw data for record
Datum *datum[] // loaded data

datum[0] = convertToInteger(raw[0]);
datum[1] = convertToFloat(raw[4]);
datum[2] = convertToShort(raw[8]);
datum[4] = convertToFloat(raw[10]);
```