Database Storage

Resources

- Architecture of a Database System (Chapter 5)
  https://dsf.berkeley.edu/papers/fntdb07-architecture.pdf

- Postgres documentation
  https://www.postgresql.org/docs/9.0/storage-page-layout.html

- Oracle documentation
  https://docs.oracle.com/cd/E11882_01/server.112/e40540/physical.htm#CNCPT1389

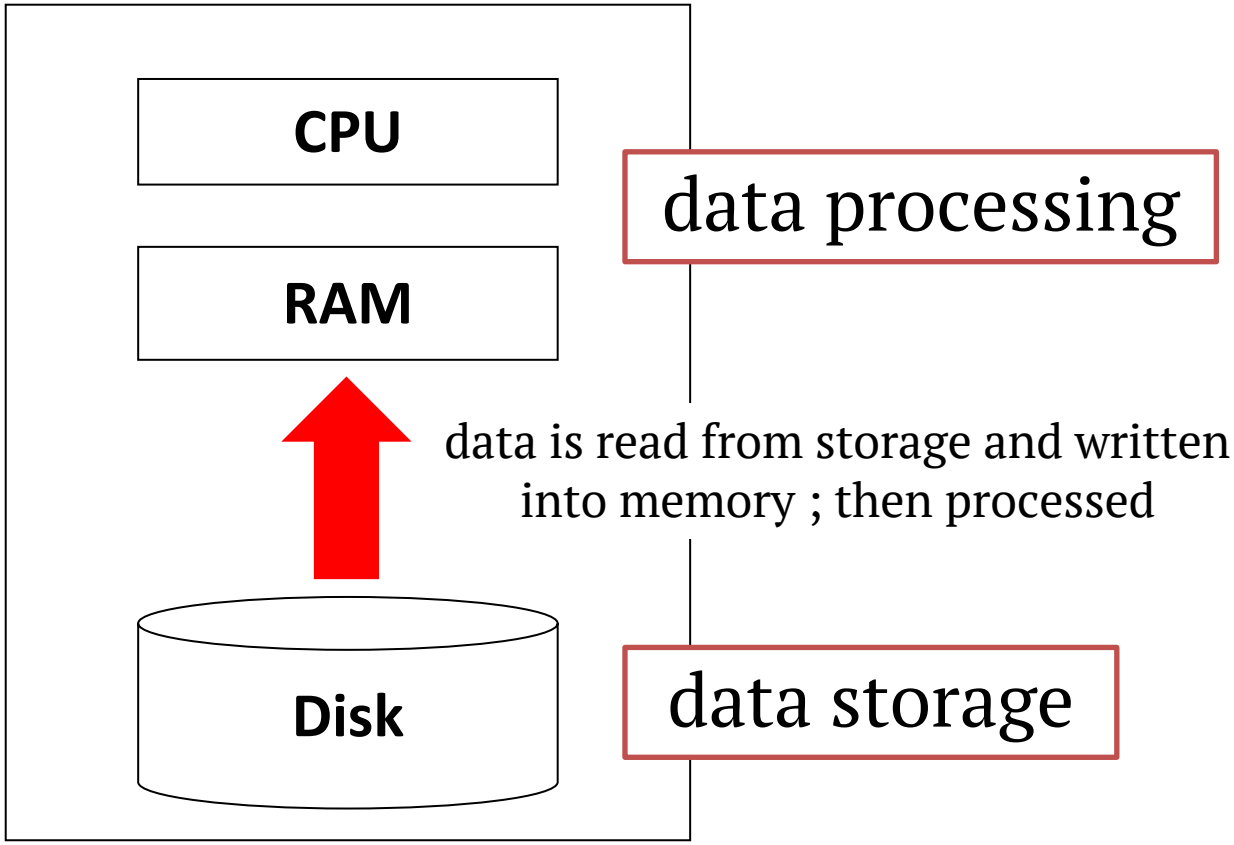
Destiny of Data : Queries

- What happens when we run a query ?
- Are all queries “equal” ?
- Are all systems good at answering queries ?

What happens when we run a query ?

- Well, the data is read and the query evaluated
- Where is data read from ?
  - Disk
    - Data is persistent
    - It may not fit in memory (but there are exceptions)
- Where is the query computed ?
  - CPU
    - At query time, data moves “up” from disk to CPU registers

What happens when we run a query?

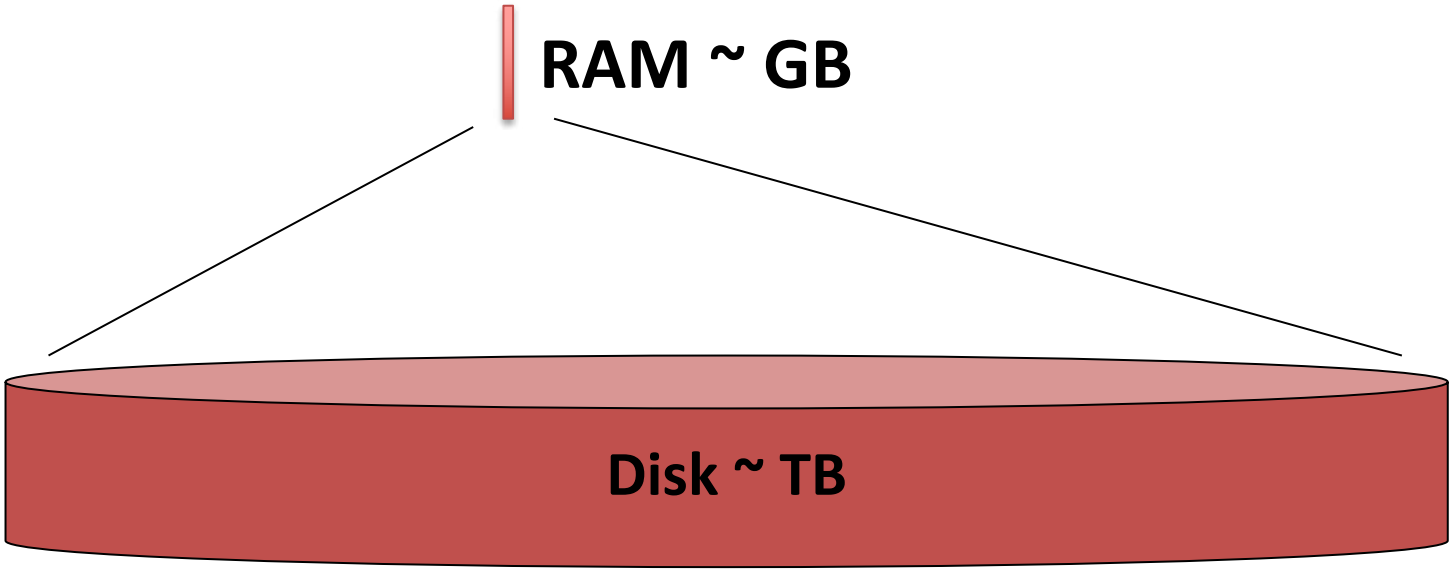


Memory : the state of affairs

sources : https://jcmit.net/memoryprice.htm https://jcmit.net/diskprice.htm

	Speed (Read/Write)	Cost/MB	
Cache	L1 read **3**TB/s	~1000$/GB	fastest and most costly storage; volatile; managed by computer hardware
RAM	DDR4 read ~**25**GB/s	~10$/GB	~100x slower & cheaper than cache
Disk	SSD read ~**0.5**GB/s	~0.2$/GB	Primary medium for the long-term storage of data

Memory : the state of affairs



- Data may not fit in memory, and DBMS architecture is specifically designed to account for this

The Query-Evaluation “Game”

- Compute answers to queries on :
  - (Possibly large) volumes of data stored on disk
  - Limited (but fast) memory

  Within a useful time (useful for the user/application)

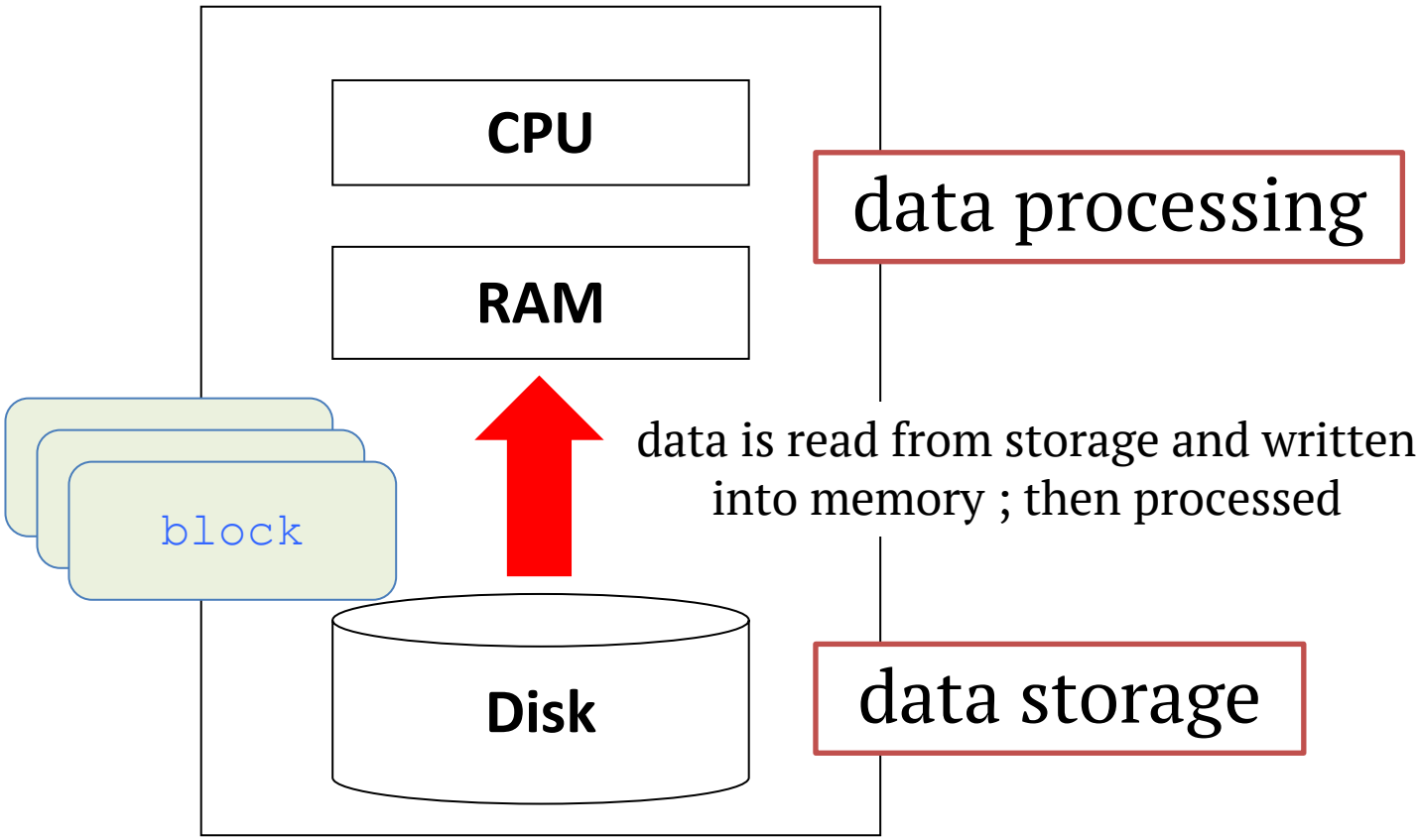
- To “win the game”, one needs to devise a strategy for :
  - **Organizing** data
  - **Moving** data from disk to memory
  - **Optimizing** query computation

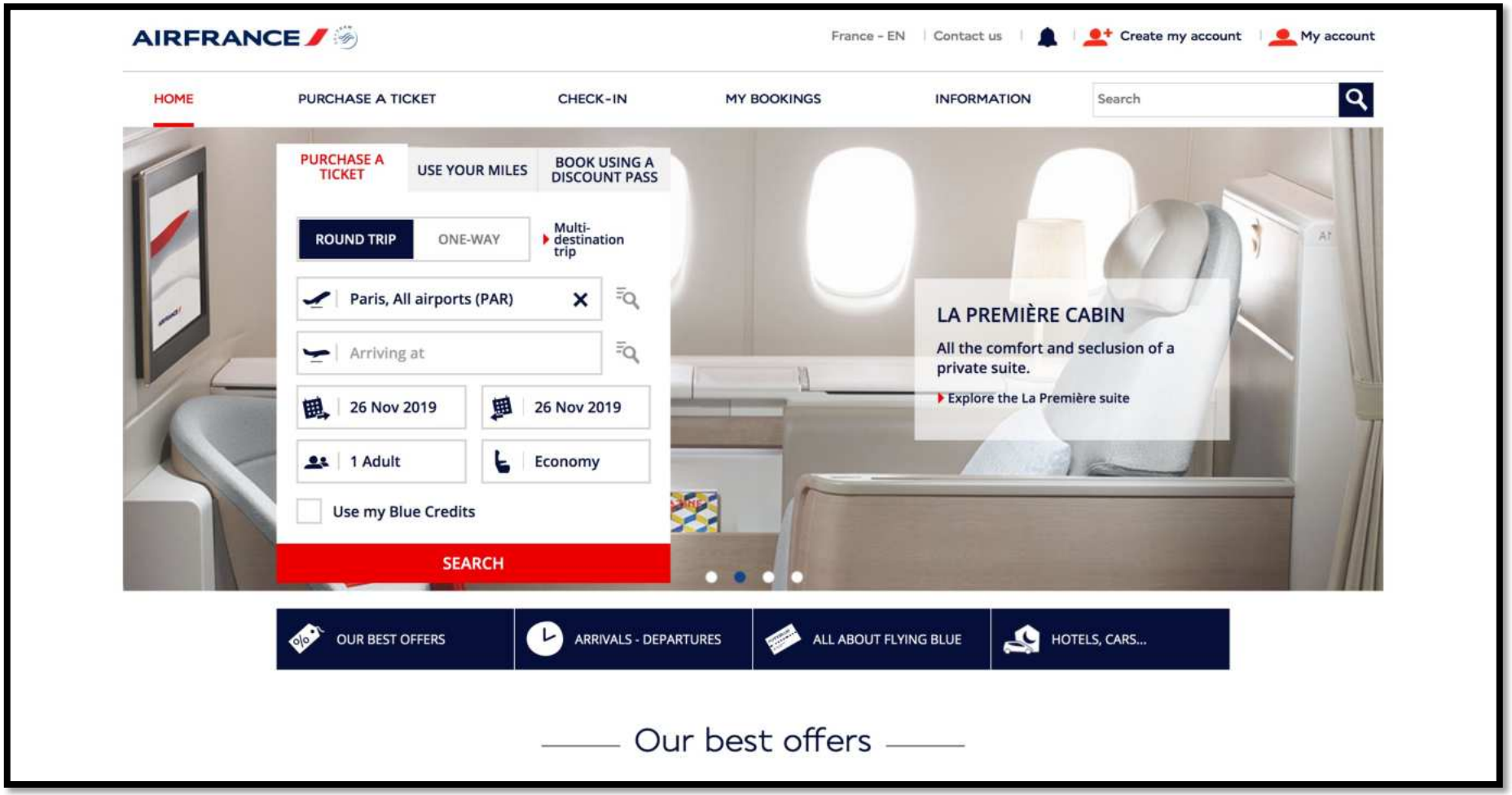
So what is Postgres doing ?

https://link.springer.com/content/pdf/bbm%3A978-1-4302-0018-5%2F1.pdf

- Postgres stores table data in multiple ***files***.
  - each file can grow up to 1GB (design choice of the Postgres system)
- A file stores a set of database **records**.
- Records are partitioned into *fixed-length storage units* called **blocks**.
  - default size (tunable) : 8KB (maximum Postgres 32K)
  - each block-id have a 32-bit integer ID (allows ~2 billion blocks)
  - max table size : #blocks x block_size (16TB to 64TB)
- **Blocks are units of both storage allocation and data transfer.**
  - Neither single records (as one may think at first), nor files are transferred from disk to memory : blocks !

What happens when we run a query?

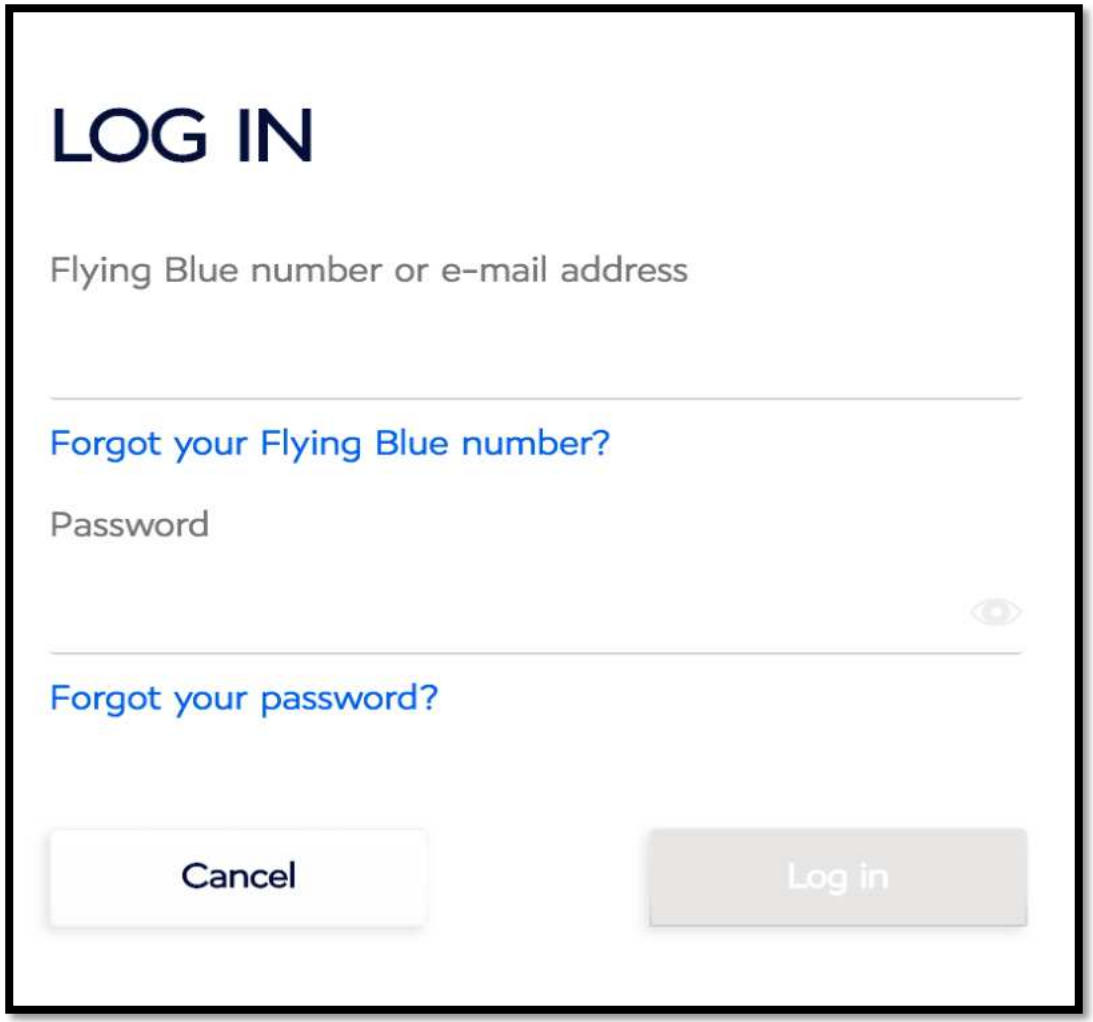




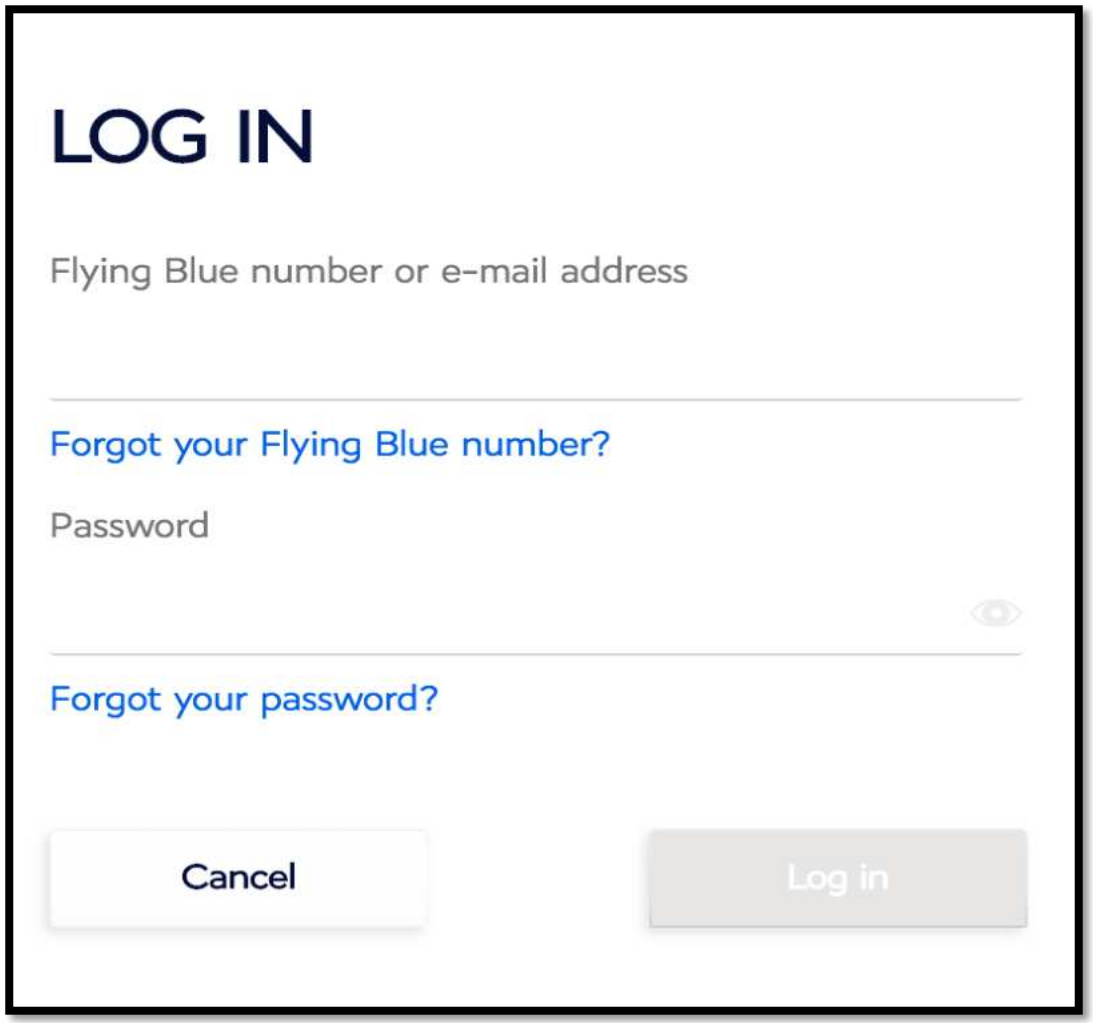
(2018) 4th european company

100+ million passengers

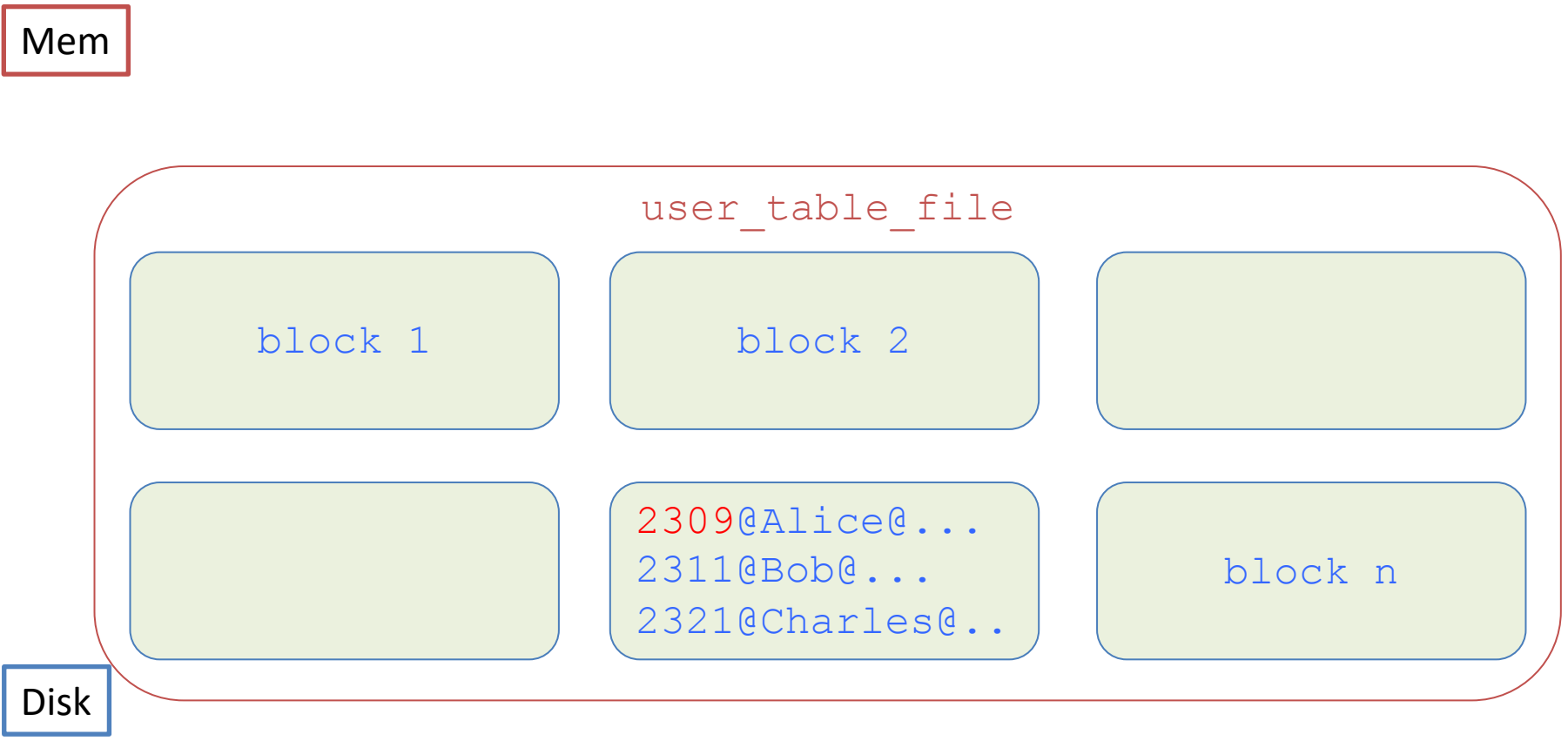
300+ destinations


LOG IN
Flying Blue number or e-mail address
Forgot your Flying Blue number?
Password
Forgot your password?
Cancel
Log in

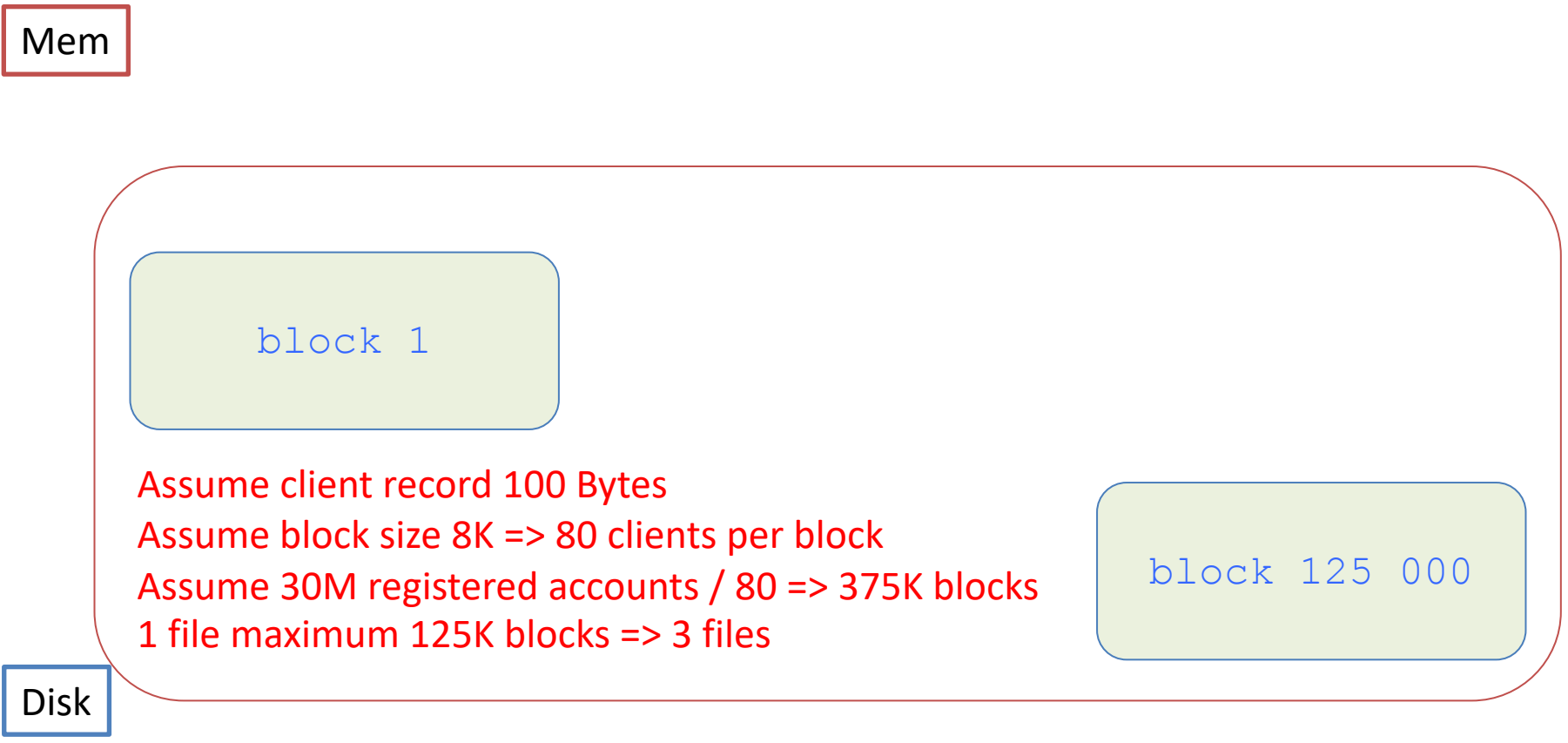
```
SELECT  *                #user profile data
FROM    users_table
WHERE   user_ID = 2309
```



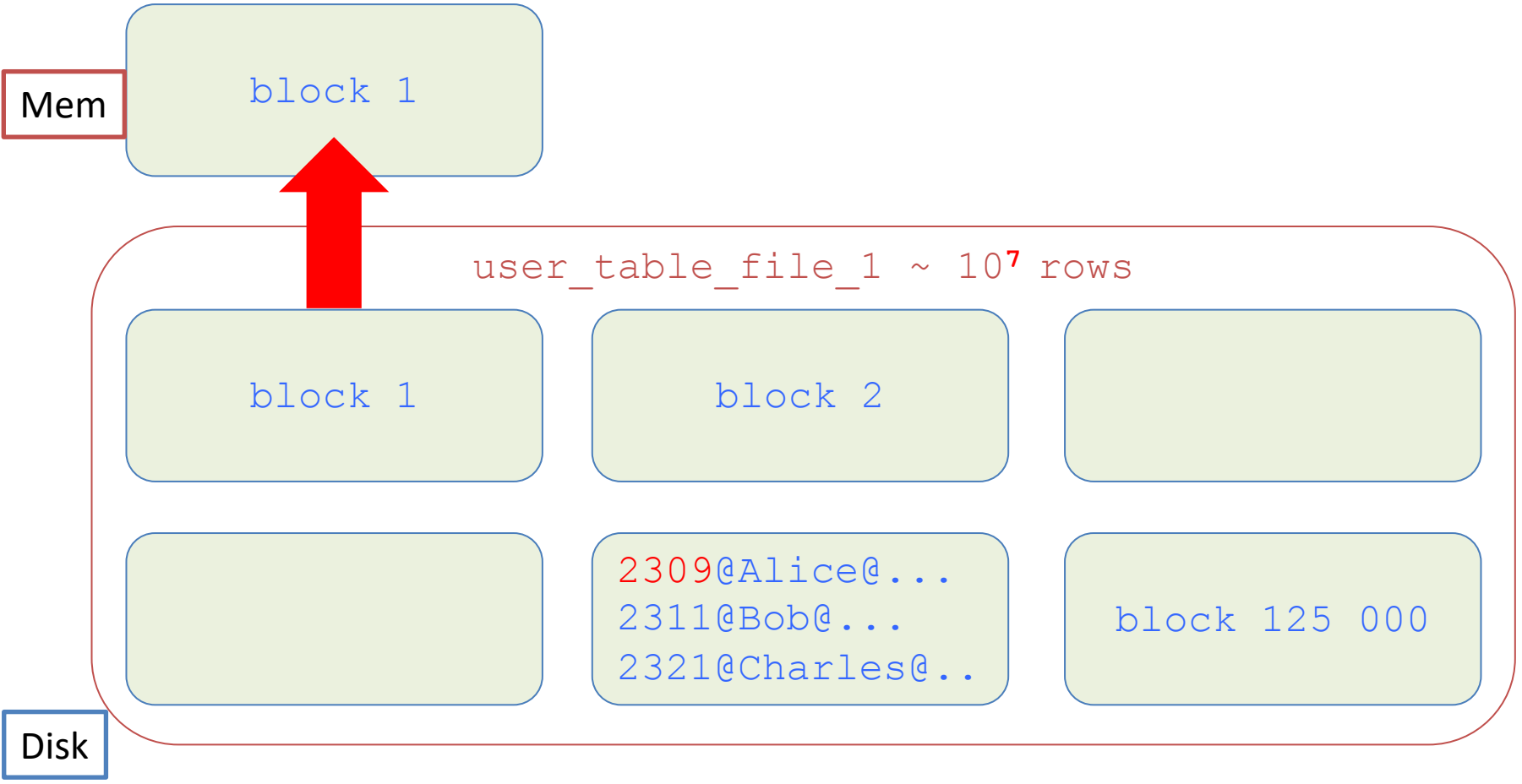
```
SELECT  *                #user profile data
FROM    users_table
WHERE   user_ID = 2309
```



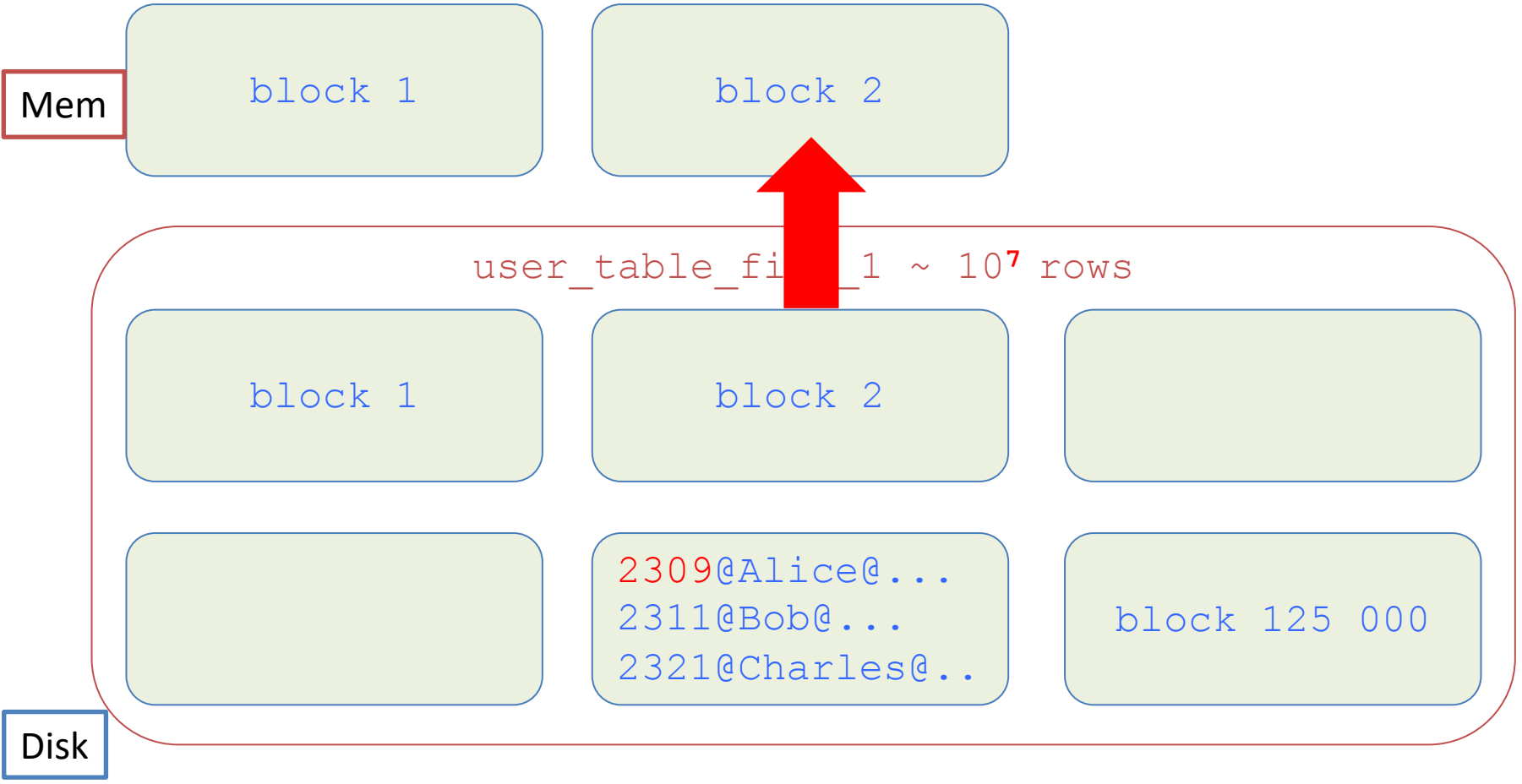
```
SELECT  *                #user profile data
FROM    users_table
WHERE   user_ID = 2309
```



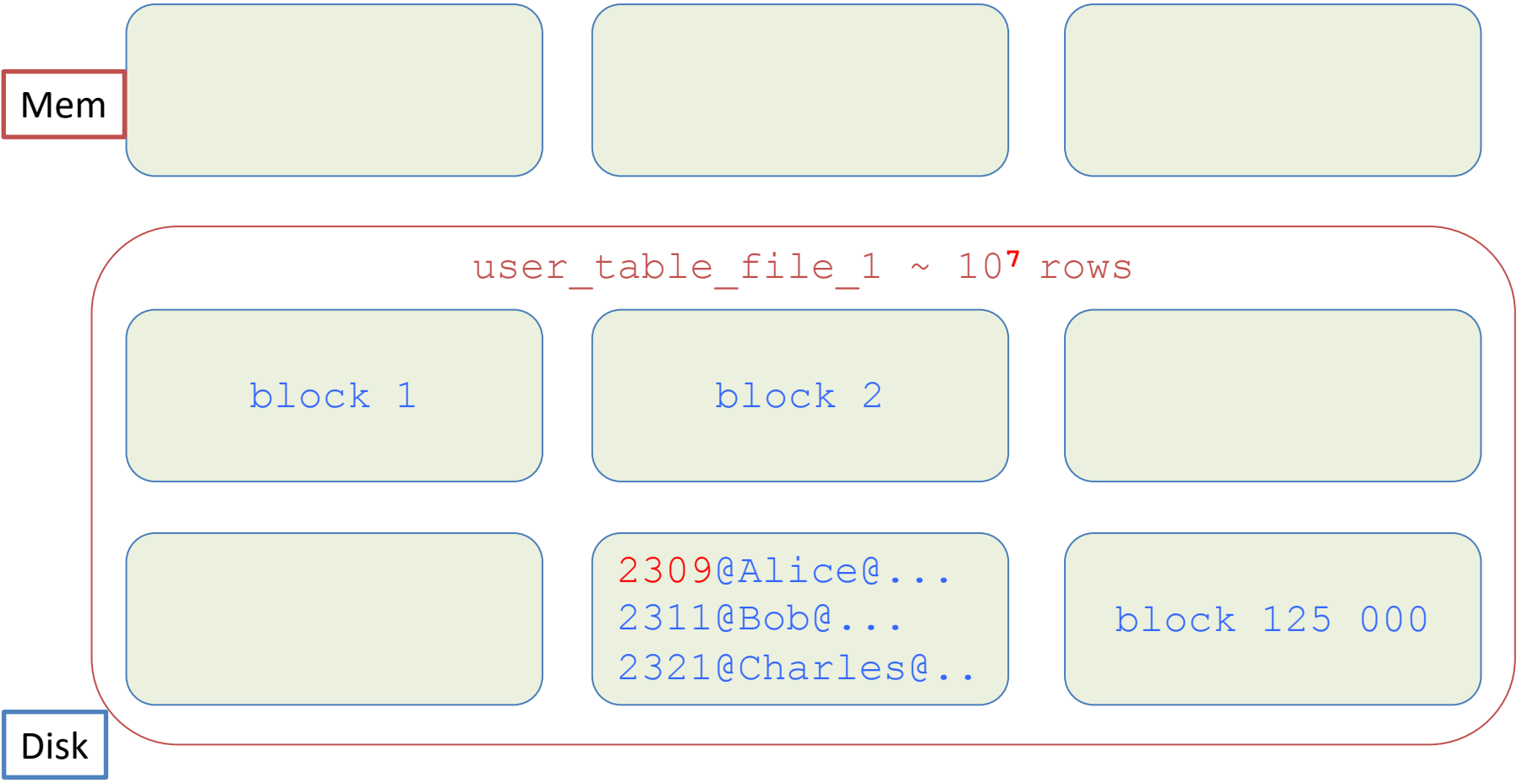
```
SELECT  *                  #user profile data
FROM    users_table
WHERE   user_ID = 2309
```



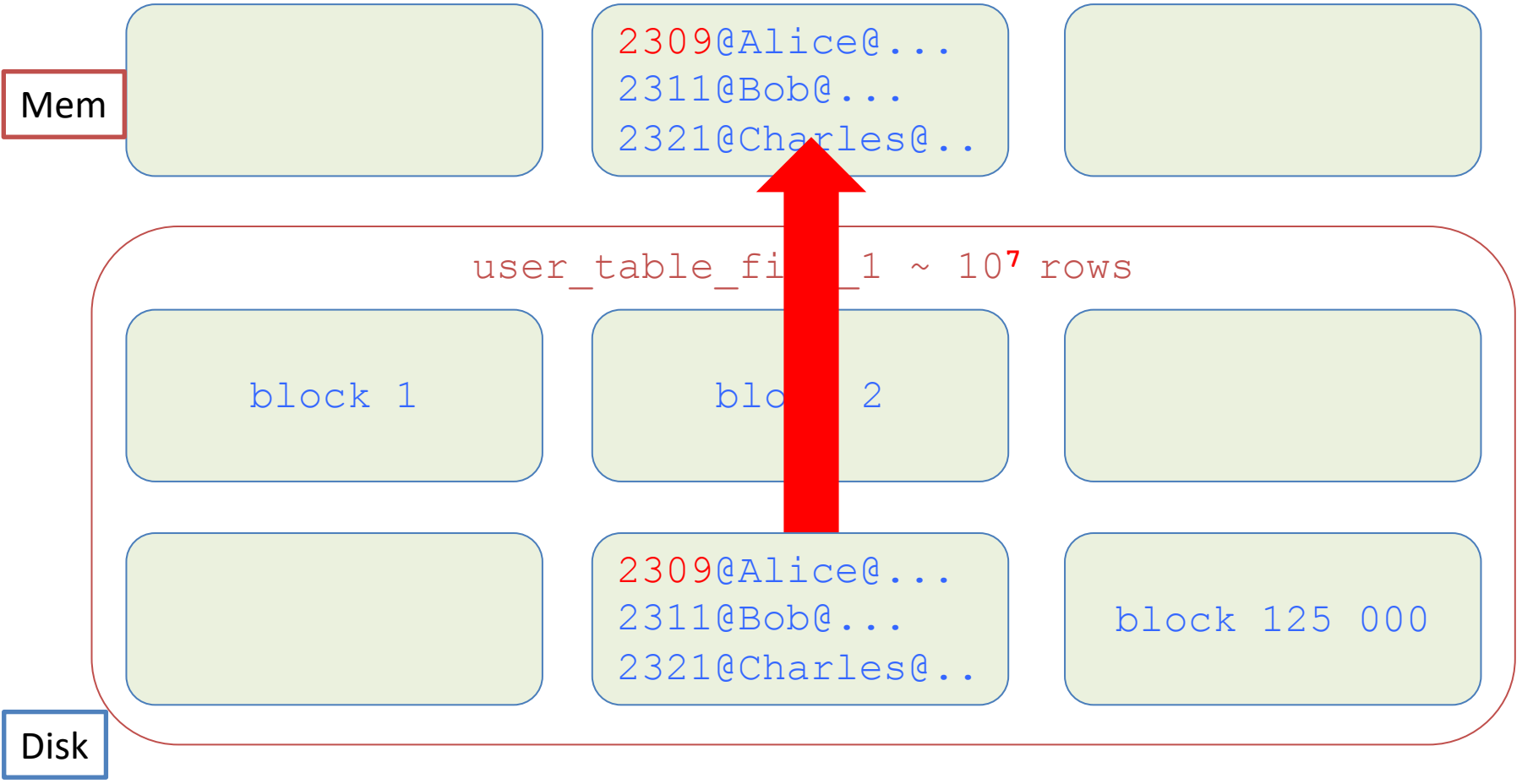
```
SELECT  *                  #user profile data
FROM    users_table
WHERE   user_ID = 2309
```



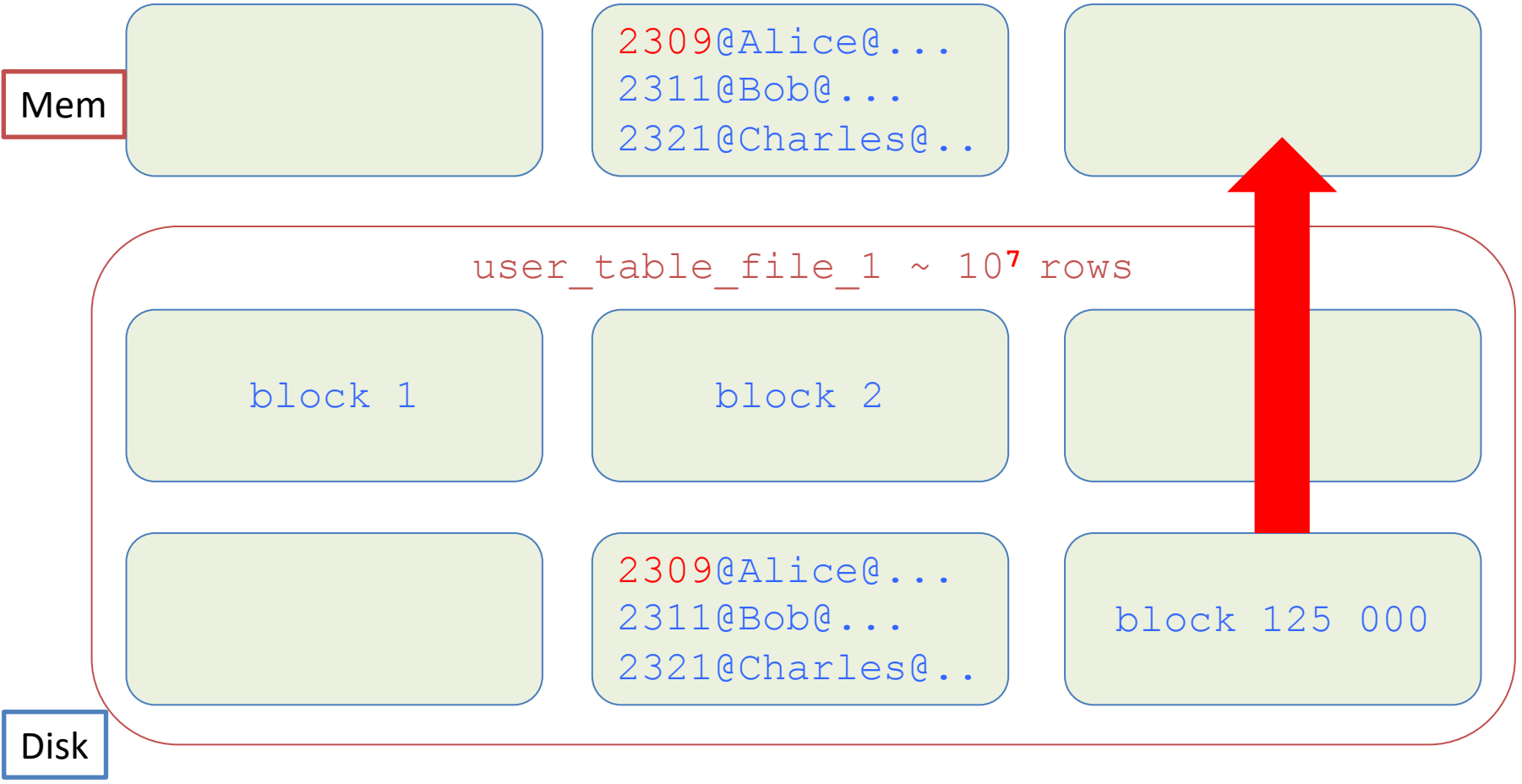
```
SELECT  *                #user profile data
FROM    users_table
WHERE   user_ID = 2309
```



```
SELECT  *                  #user profile data
FROM    users_table
WHERE   user_ID = 2309
```



```
SELECT  *                  #user profile data
FROM    users_table
WHERE   user_ID = 2309
```



```
SELECT  *                #user profile data
FROM    users_table
WHERE   user_ID = 2309
```

```
SELECT  *               #user profile data
FROM    users_table
WHERE   user_ID = 2309     #PK
```

In reality DB use indexes !!

Mem

index
ROW for **user_ID** = 2309
block 7459 offset 45

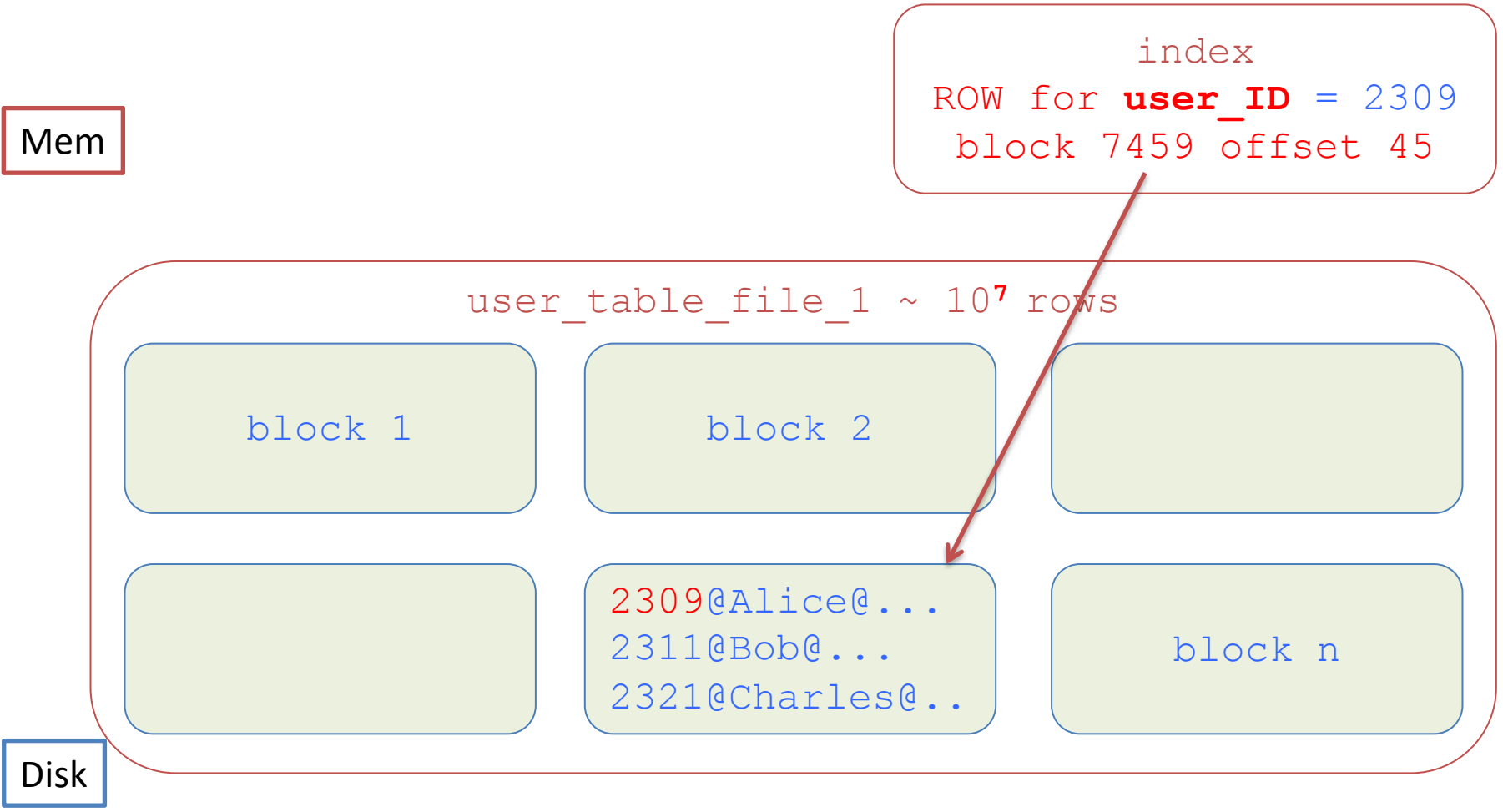
user_table_file_1 ~ 10^7 rows

block 1	block 2	
	2309@Alice@... 2311@Bob@... 2321@Charles@..	block n

Disk

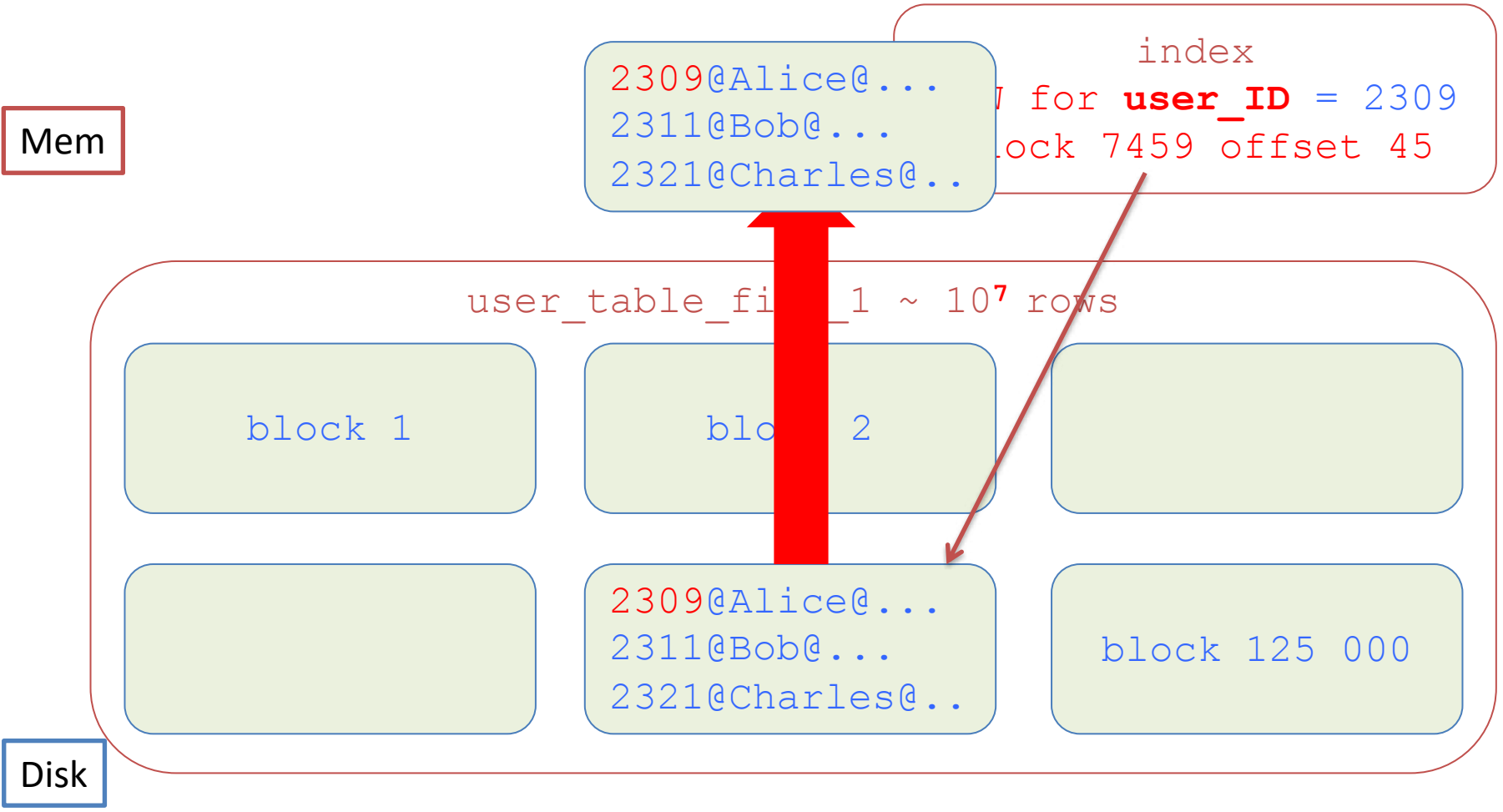
```
SELECT  *               #user profile data
FROM    users_table
WHERE   user_ID = 2309     #PK
```

In reality DB use indexes !!



```
SELECT  *                  #user profile data
FROM    users_table
WHERE   user_ID = 2309     #PK
```

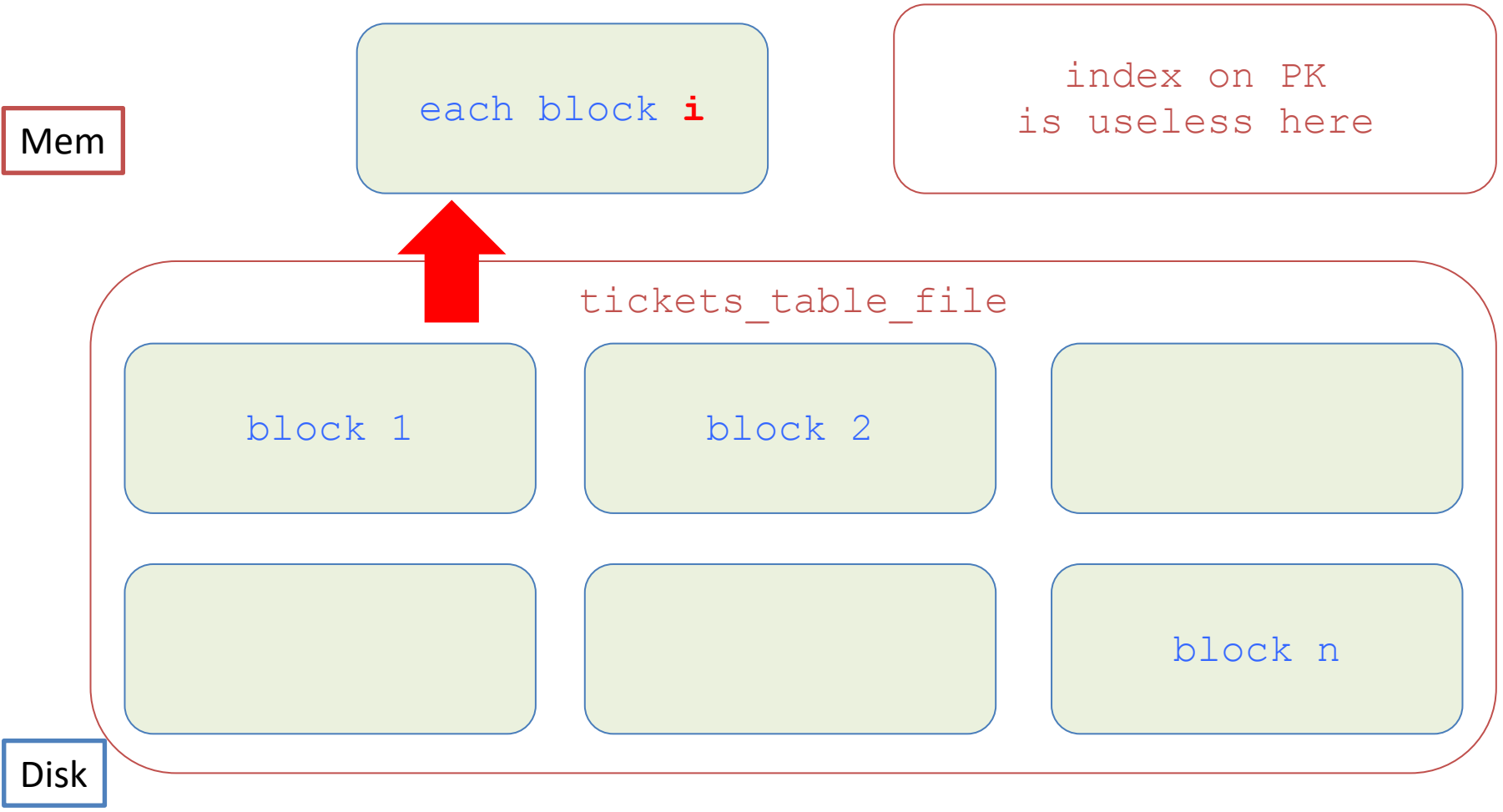
In reality DB use indexes !!



```
SELECT    SUM(price)
FROM      tickets_table
WHERE     year = NOW.year
```

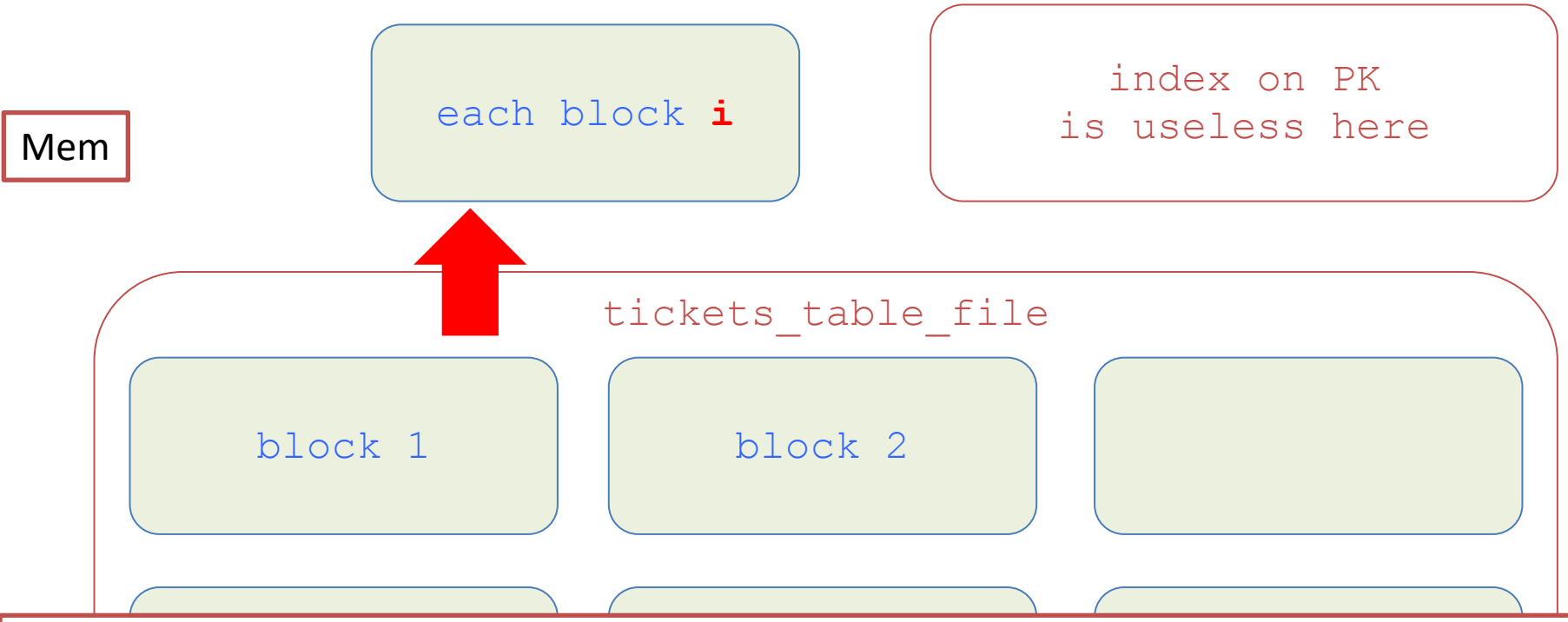
```
SELECT  SUM(price)
FROM    tickets_table
WHERE   year = NOW.year
```

Analytics case



```
SELECT  SUM(price)
FROM    tickets_table
WHERE   year = NOW.year
```

Analytics case



Quiz : how many files and blocks/year with a 40 Byte ticket record in Postgres ? Assume 100M tickets/year.

How many files and blocks/year with a 40 Byte ticket record in Postgres ? Assume 100M tickets/year.

- Postgres
  - Block size = 8K
  - 1 file maximum 125K blocks

- 1 ticket = 40Bytes
- 100M tickets per year

- Storage cost :
  - 40Bytes x 100M = 4 000 M per year
  - 4000 M / 8 K = 500M blocks
  - 500K / 125K = 4 fichiers

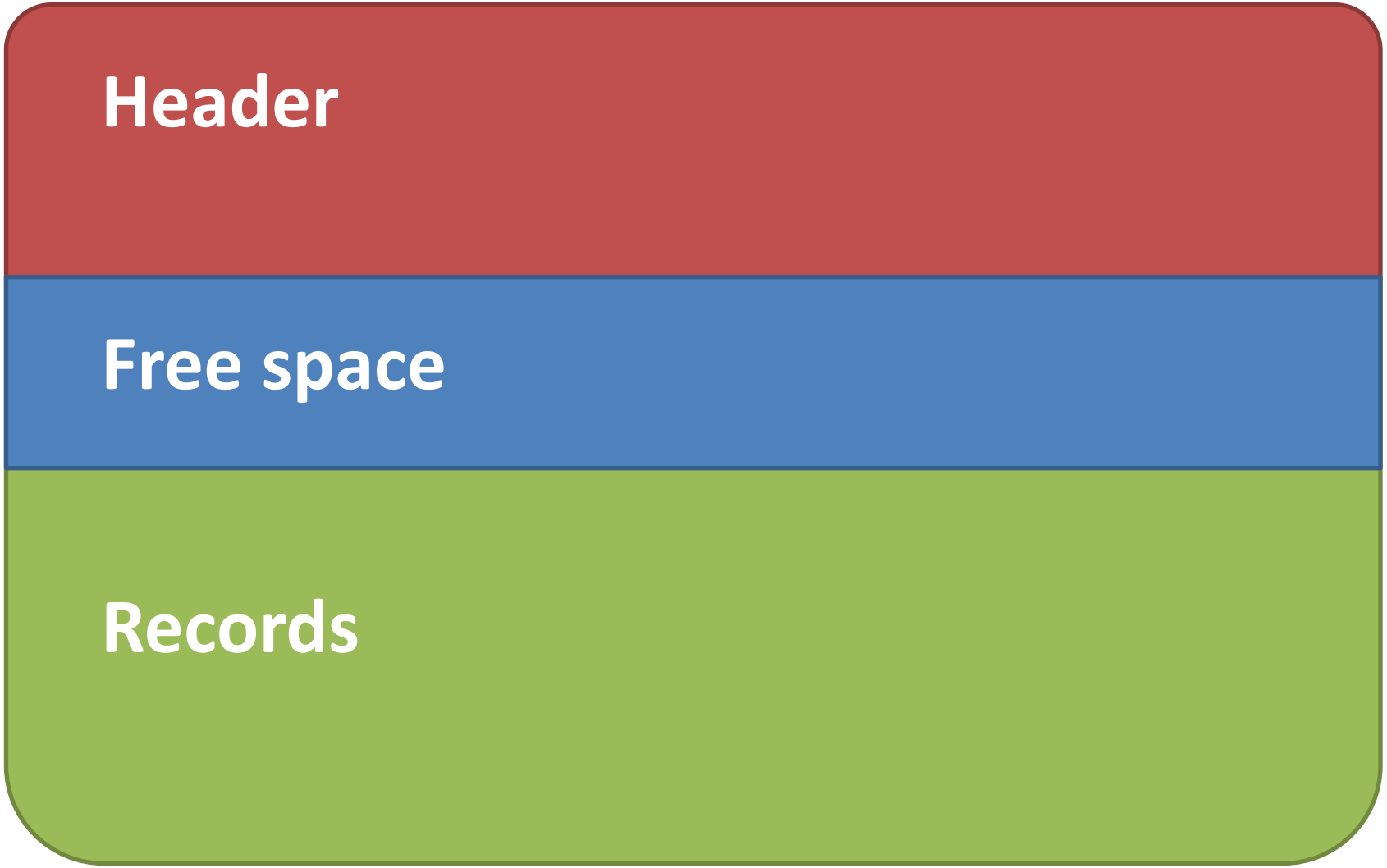
The Query-Evaluation “Game”

- **Blocks are units of both storage allocation and data transfer.**
  - Neither single records (as one may think at first), nor files are transferred from disk to memory : **blocks** !

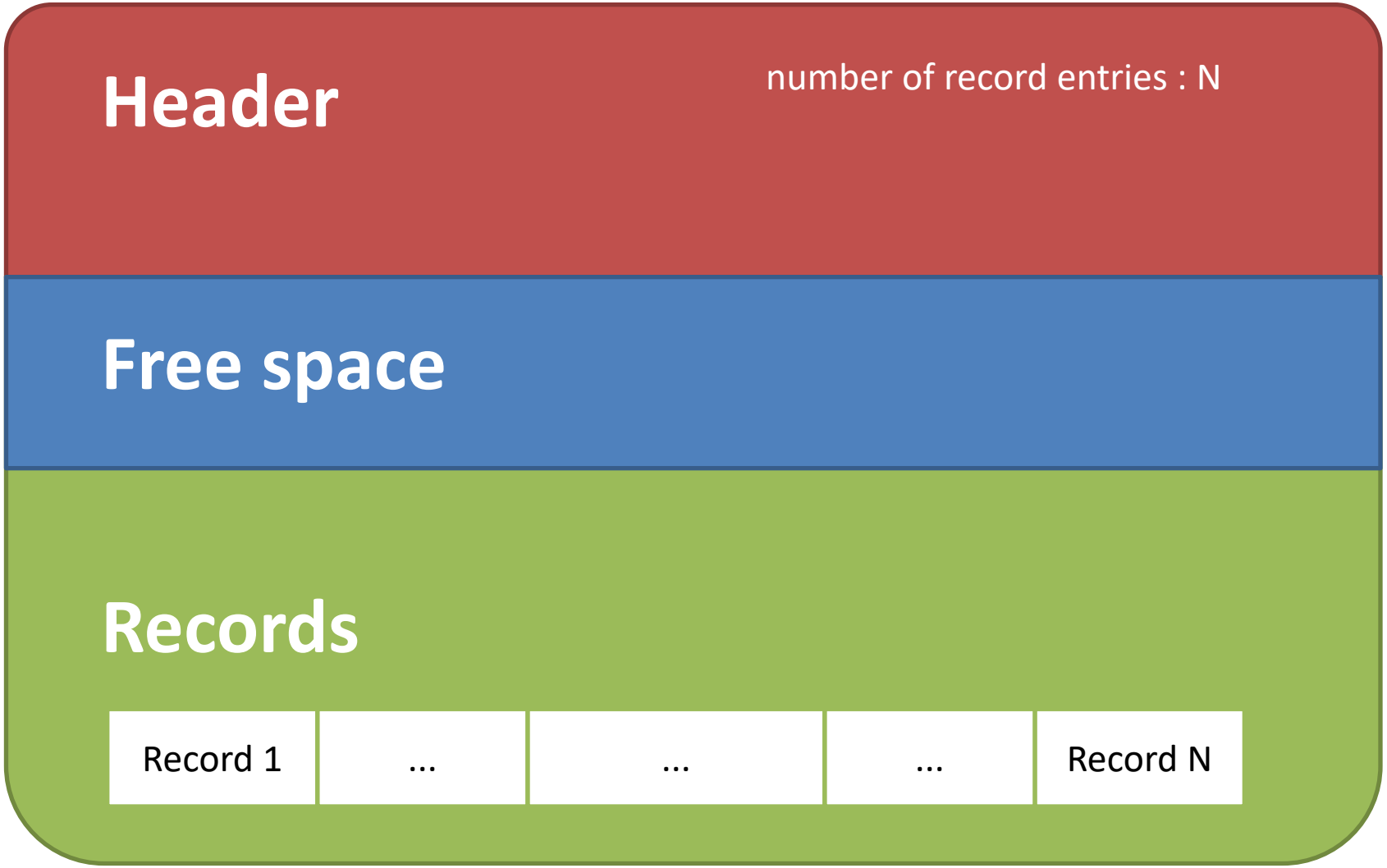
The Query-Evaluation “Game”

- To “win the game” the DB seeks to minimize the number of block transferred from disk to memory
  - avoid loading a block twice
  - avoid loading useless blocks
  - keep as many blocks as possible in main memory
    - Locality principle
  - reduce the number of disk accesses

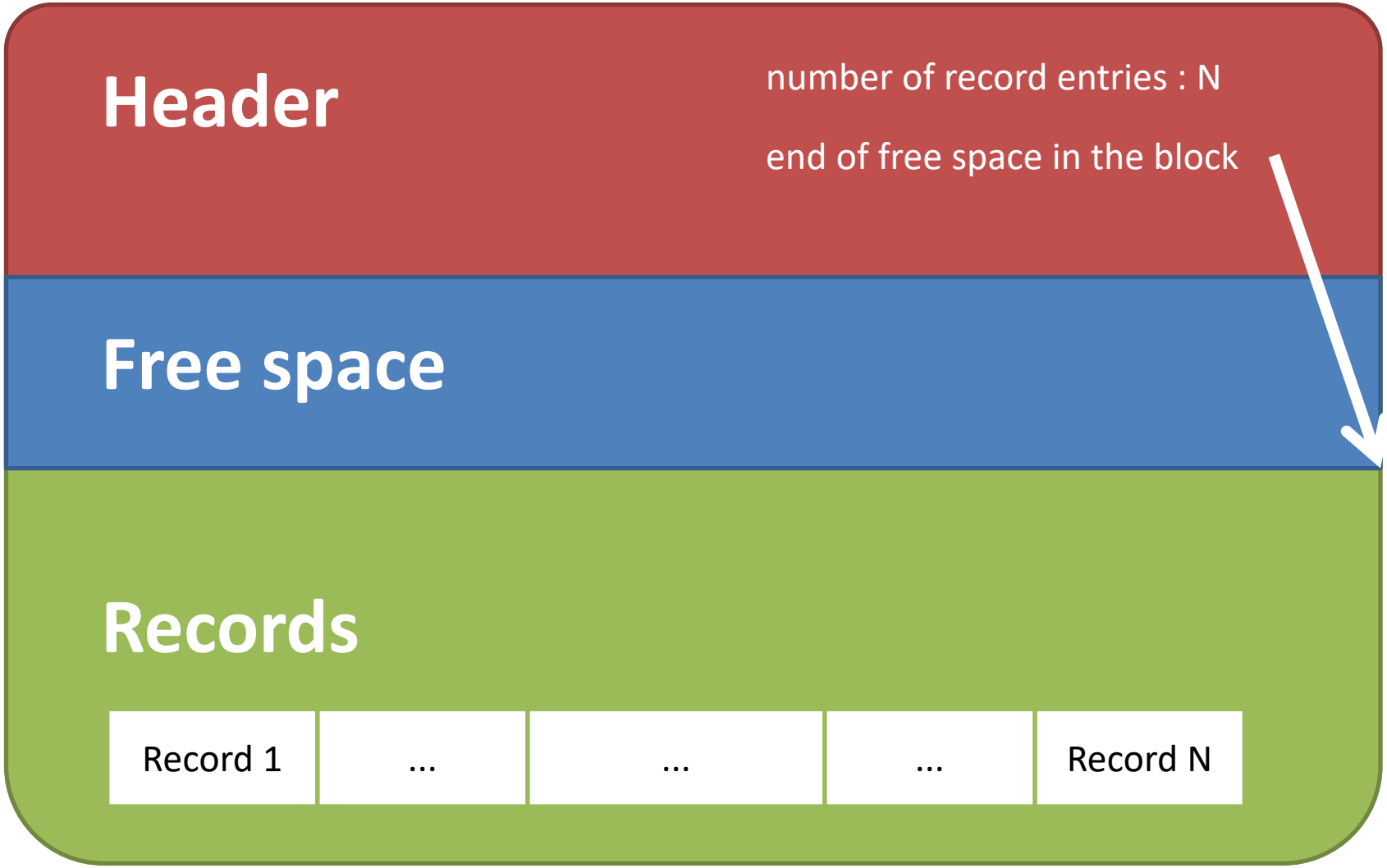
Block organization



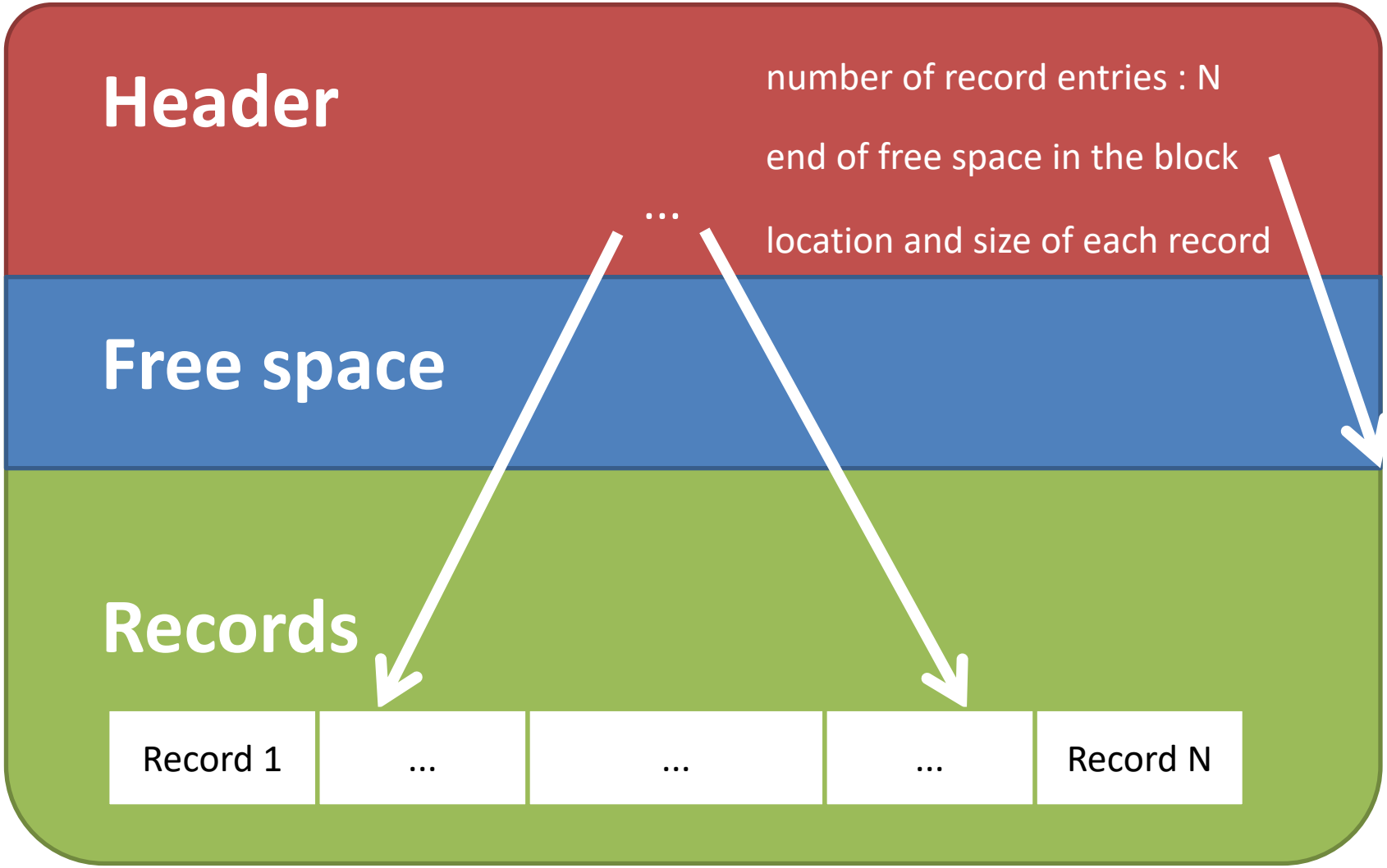
Block organization



Block organization

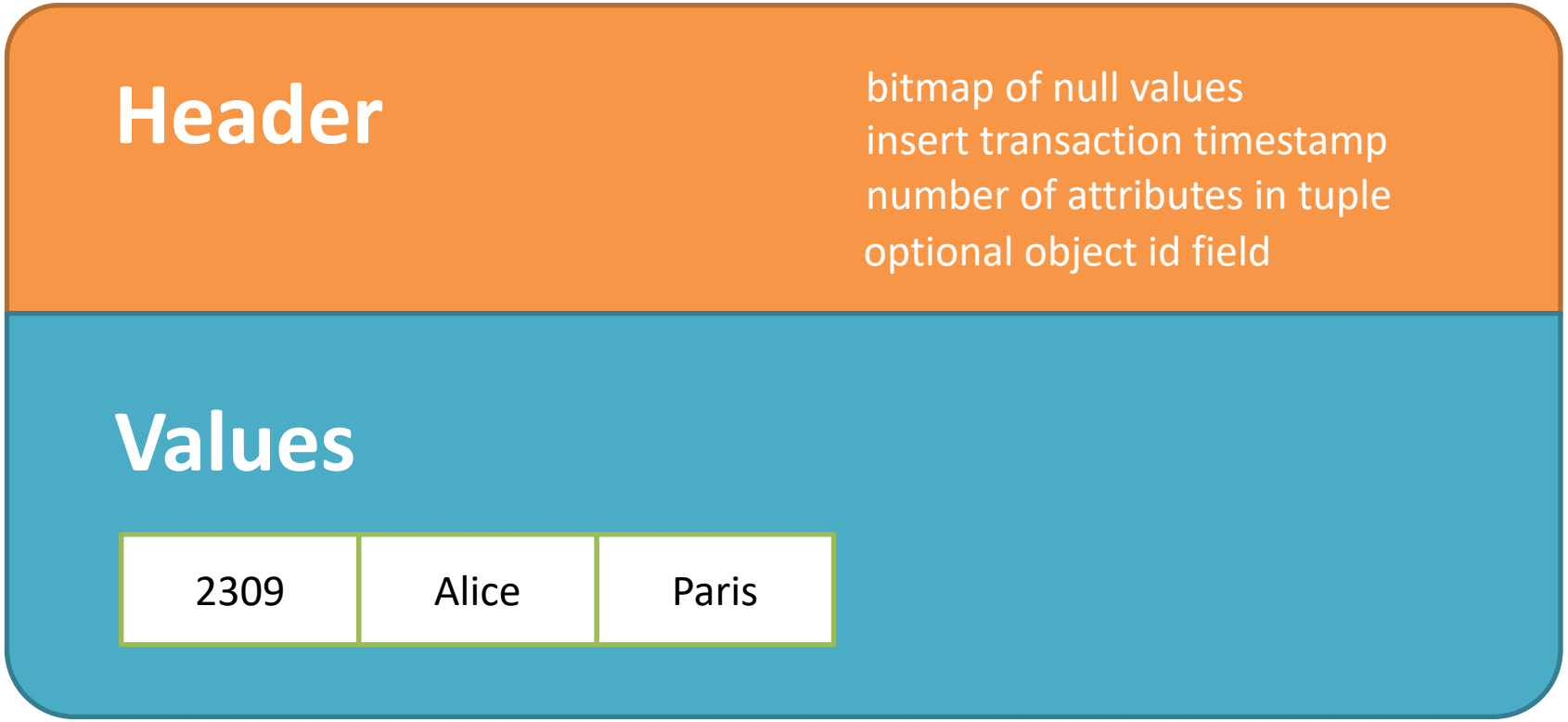


Block organization



Record organization

https://www.postgresql.org/docs/9.0/storage-page-layout.html

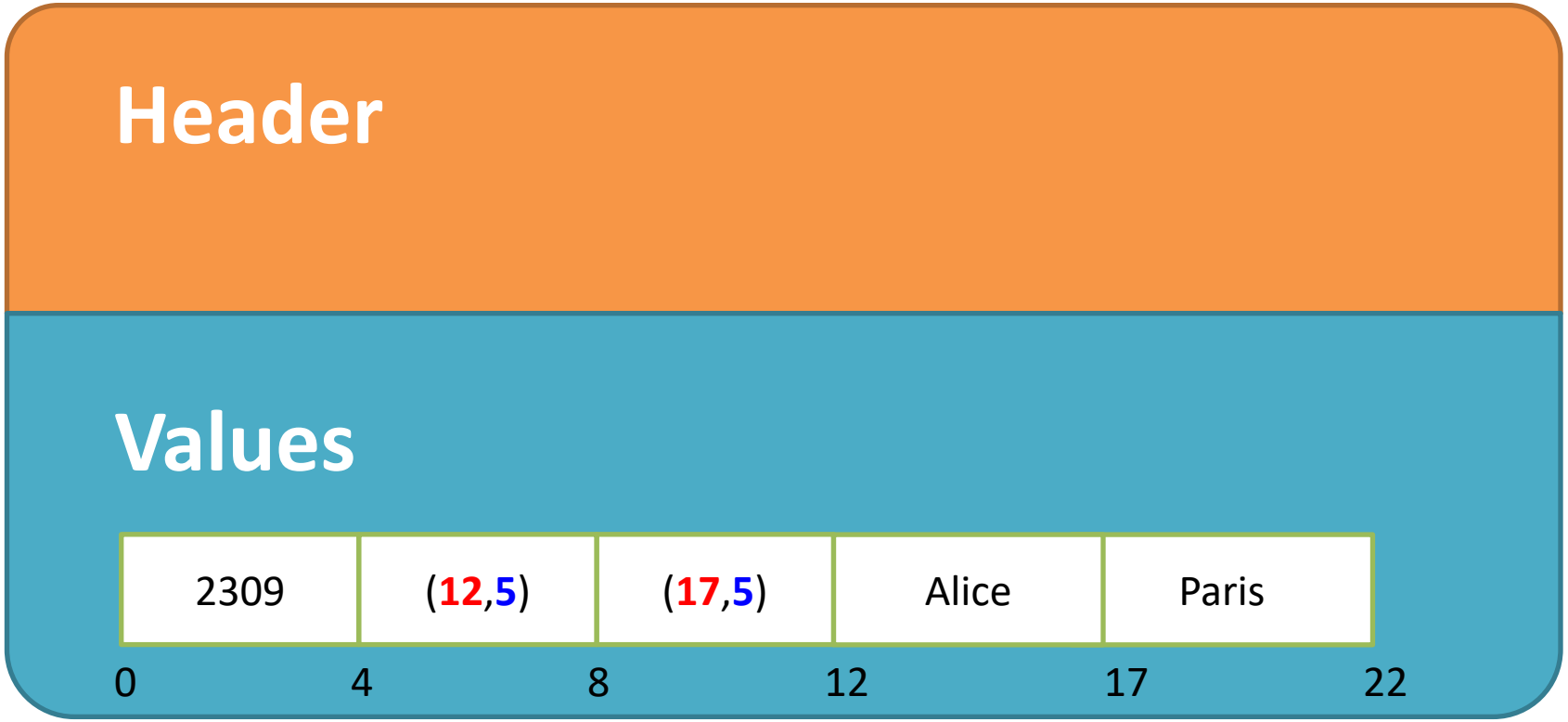


- Postgres : fixed-size header (~23 bytes), followed by optional null bitmap, optional object ID field, user data

Record organization

- Most of records are variable length
  - they occur as soon as one uses the ***varchar*** type
- Attributes are stored in order
  - Following the CREATE TABLE statement
- Variable length attributes can be represented by fixed size (offset, length), with actual data stored after all fixed length attributes
  - Efficient for searching a field in the middle of the row

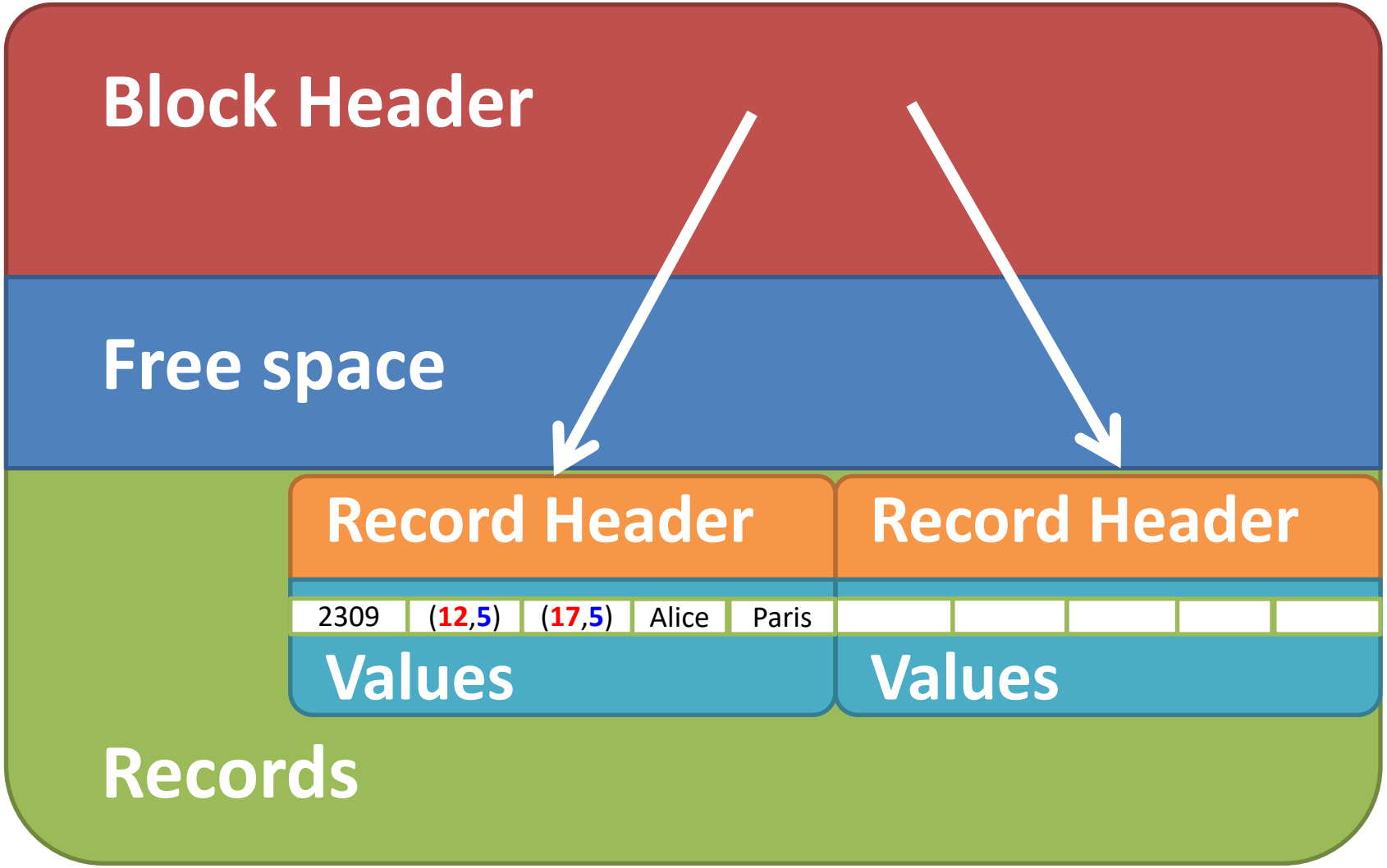
Record organization



- Variable length attributes represented by fixed size
(offset, length)

with actual data stored after all fixed length attributes

Summing up

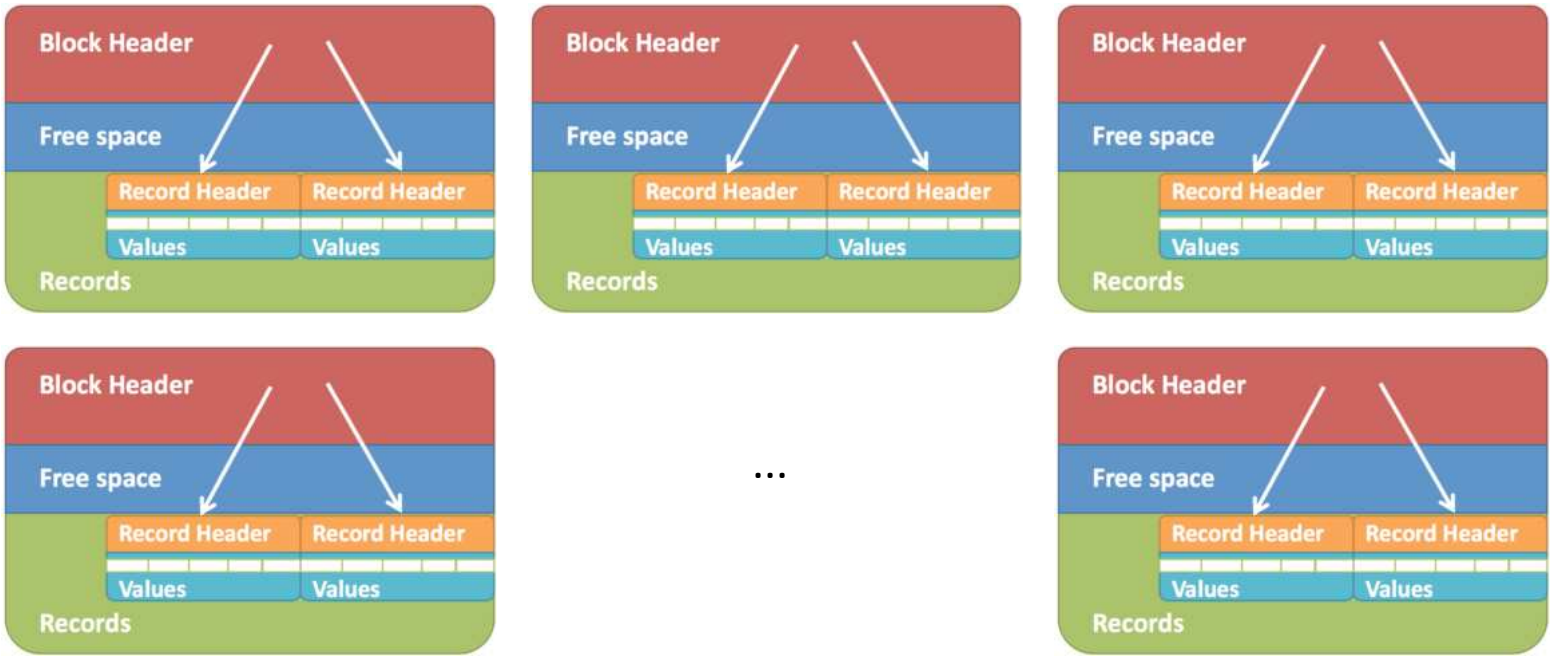

Block Header
Free space
Record Header
Record Header
2309
(12,5)
(17,5)
Alice
Paris
Values
Values
Records

Summing up

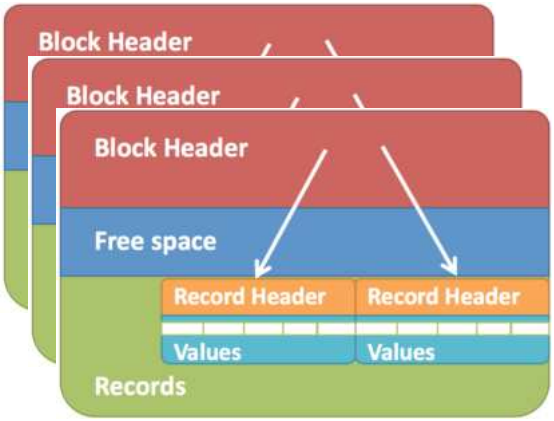
https://www.sqlite.org/fileformat.html#:~:text=The%20first%20100%20bytes%20of%20the%20database%20file%20comprise%20the,first%20(big%2Dendian).

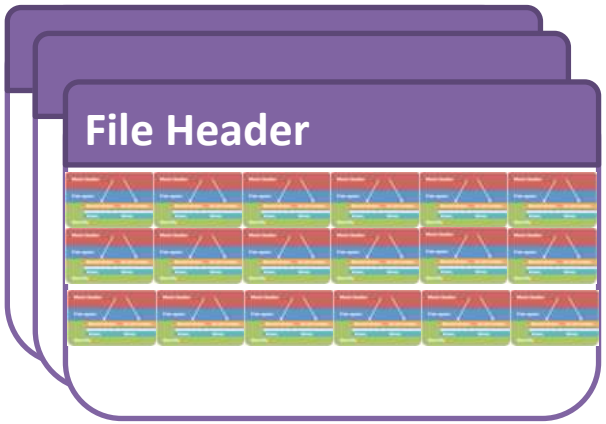
File Header

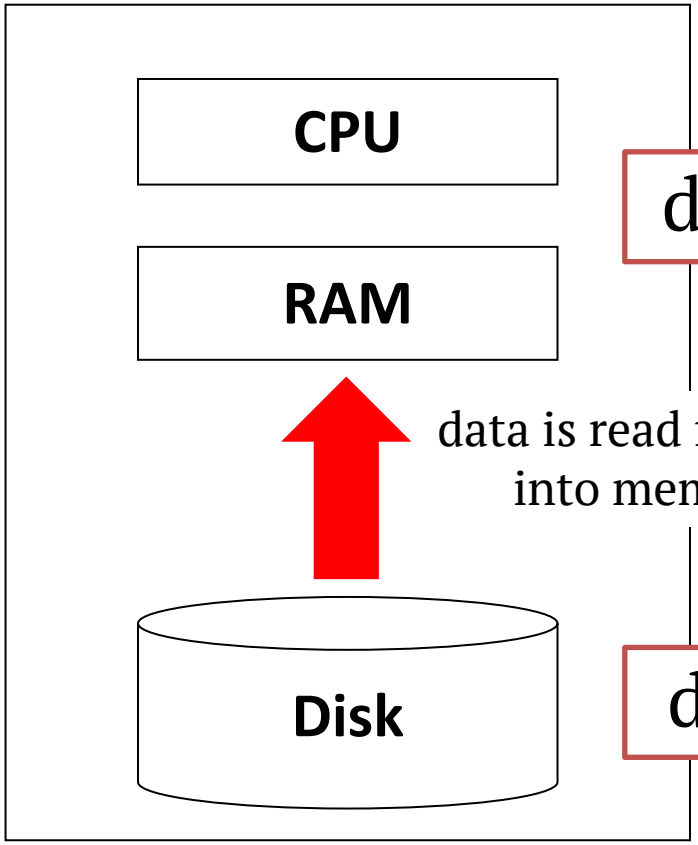
file type (DBMS has many)
the database page size
number of free blocks ...



What happens when we run a query?







data processing

data is read from storage and written into memory ; then processed

data storage

Summing up

- Databases physical organization store records in **blocks** that are moved from disk to memory
- Performances depend on block movement
- Factors that impact block movement are :
  - Of course, DBMS architecure (system)
  - The type of query (user)
  - The relational schema design (user)
    - We will see the importance of “star-schemas”
  - Tuning (eg., indexes) (DB admin)
  - Optimizations (system)