Amr Elhelw's

TECH VAULT

DATABASE INDEXES

INTERSECTION - UNION
COVERING INDEX
CLUSTERED INDEX

Composite indexes

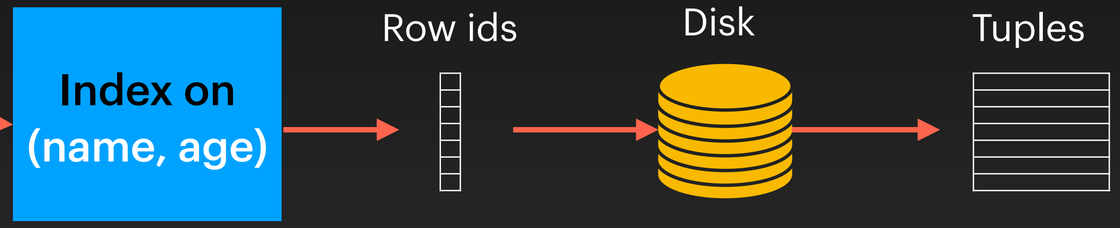
- **Queries**
  - Name = ‘John’ AND age > 30
  - Name = ‘Ann’ AND Job = ‘Manager’
- **Which indexes to build?**
  - Single-column indexes (Name), (age), (job)
  - (Name, age) works for first query
  - (Name, job) works for second query
  - (Name, age, job) or (Name, job, age)

works for first.
works for name only in second
and seq filter for job

the opposite

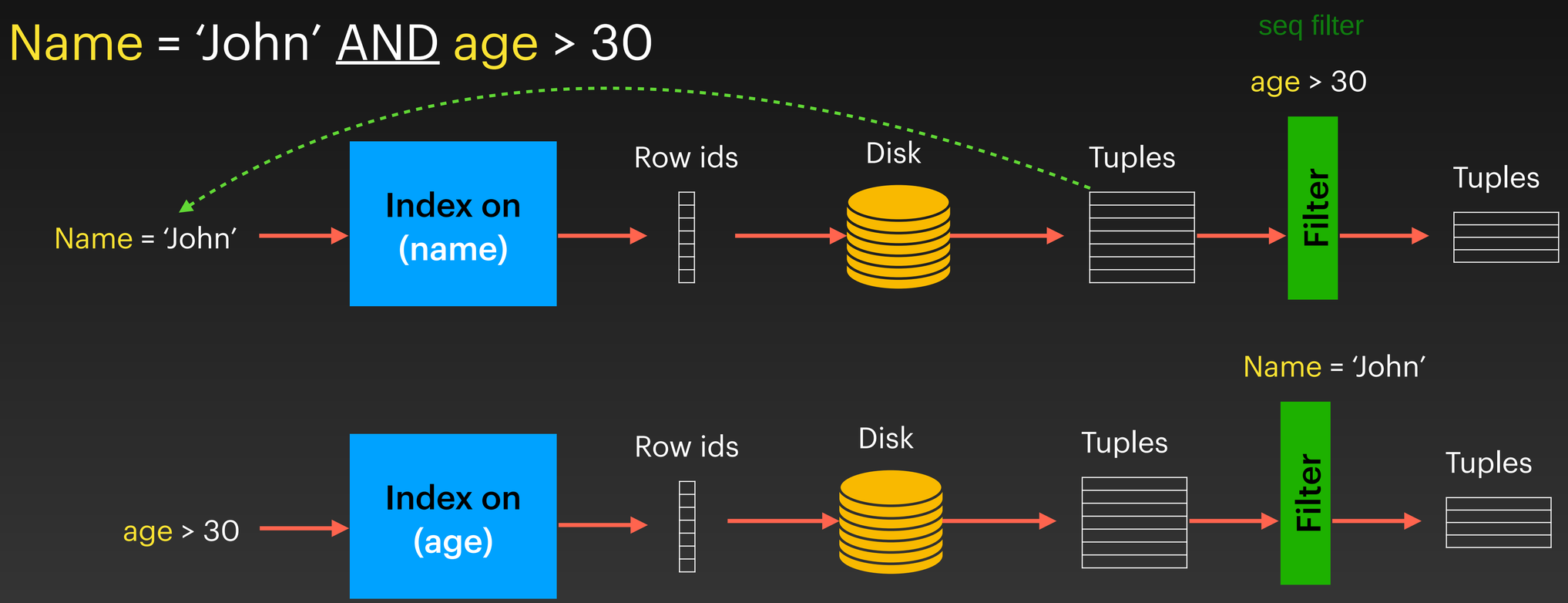
Query on 2 attributes

Name = ‘John’ AND age > 30



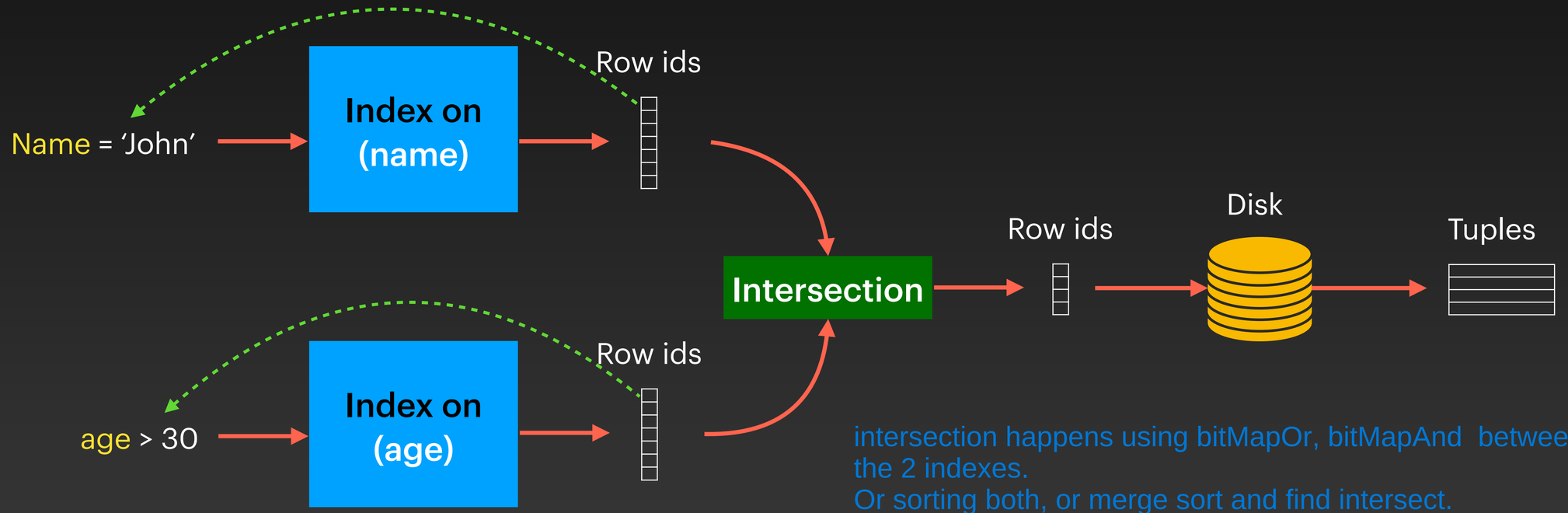
normal query

Query on 2 attributes



Query on 2 attributes

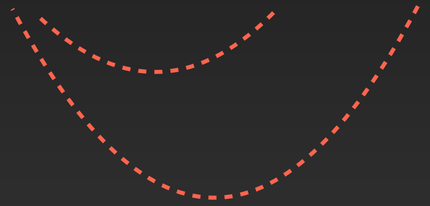
Name = ‘John’ AND age > 30



intersection happens using bitMapOr, bitMapAnd between the result of the 2 indexes.
Or sorting both, or merge sort and find intersect.
Or hashing on of results and check the other if found in hash map

Index intersection

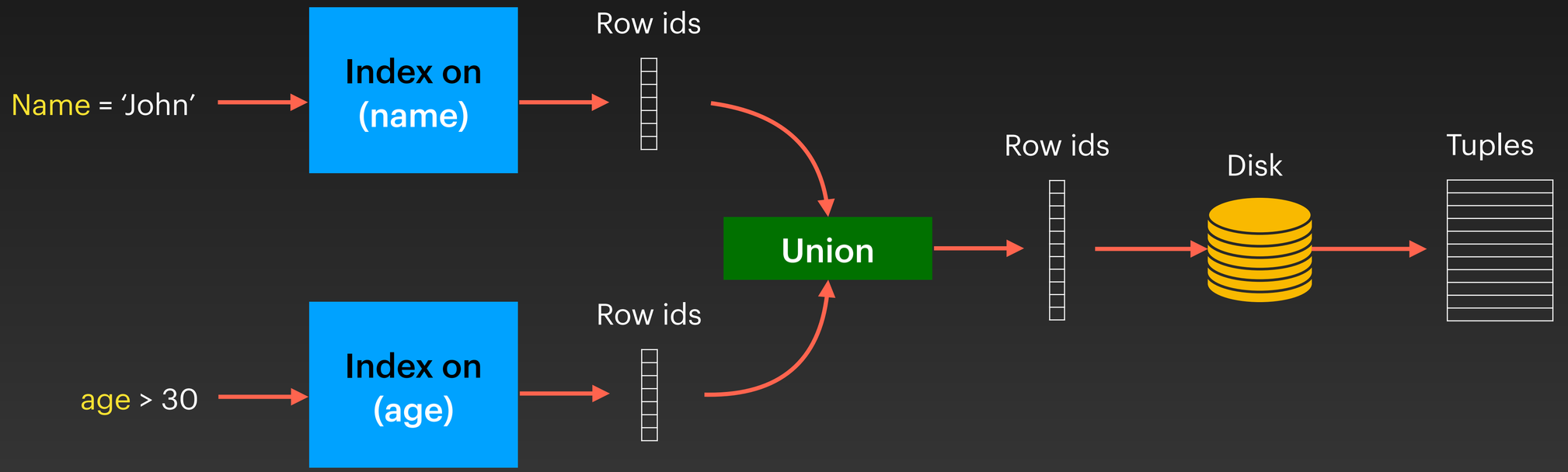
- **Queries**
  - Name = ‘John’ AND age > 30
  - Name = ‘Ann’ AND Job = ‘Manager’
- **Which indexes to build?**
  - Single-column indexes (Name), (age), (job)



before creating indexes ask yourself:
* Do your queries always get result from 1 or more specific columns or it is a combination of different columns.
* Do your queries contains Or ? so choose index on each col to help in index union
* Does your system have alot of updates?

Query on 2 attributes

Name = ‘John’ OR age > 30



Covering Index

in not covering, need to go to storage to fetach other data
even if using index first to know what to fetch from storage.

index only when using explain on a query

Index on (age)

Covering?

not ll attributes in the index

```
SELECT *
FROM person
WHERE age < 20
```

```
SELECT age
FROM person
WHERE age < 20
```

Amr Elhelw's TECH VAULT

Covering Index

Index on (name, age)

Covering?

```
SELECT *
FROM person
WHERE age < 20
AND name = ‘John’
```

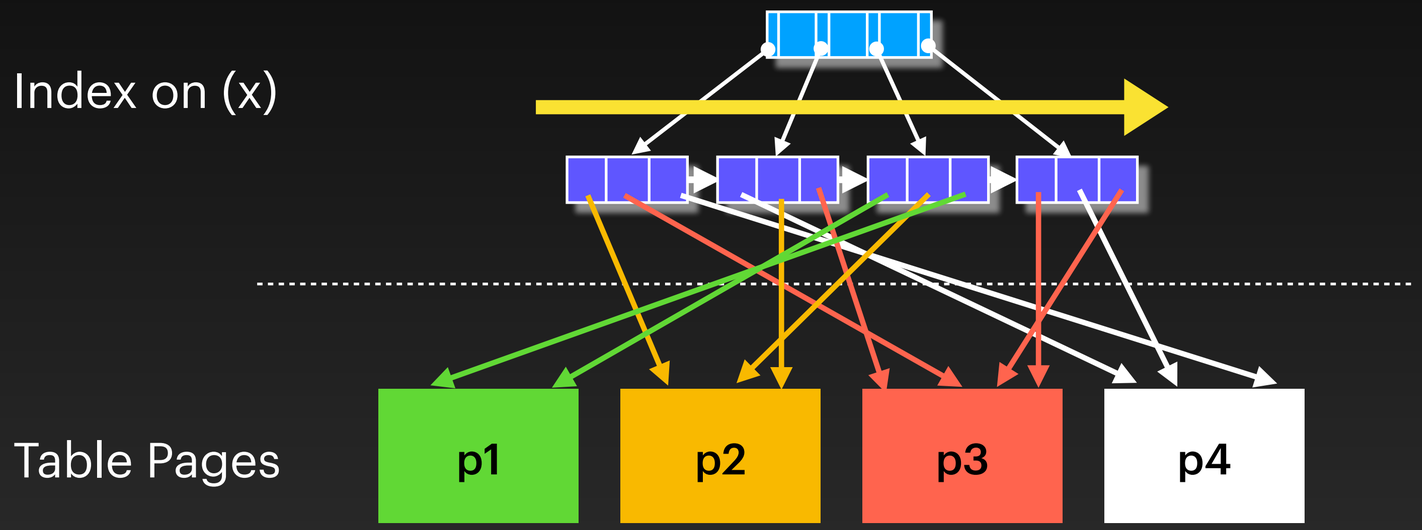
```
SELECT age
FROM person
WHERE age < 20
AND name = ‘John’
```

Covering?

```
SELECT name
FROM person
WHERE age < 20
```

```
SELECT age
FROM person
WHERE name = ‘John’
```

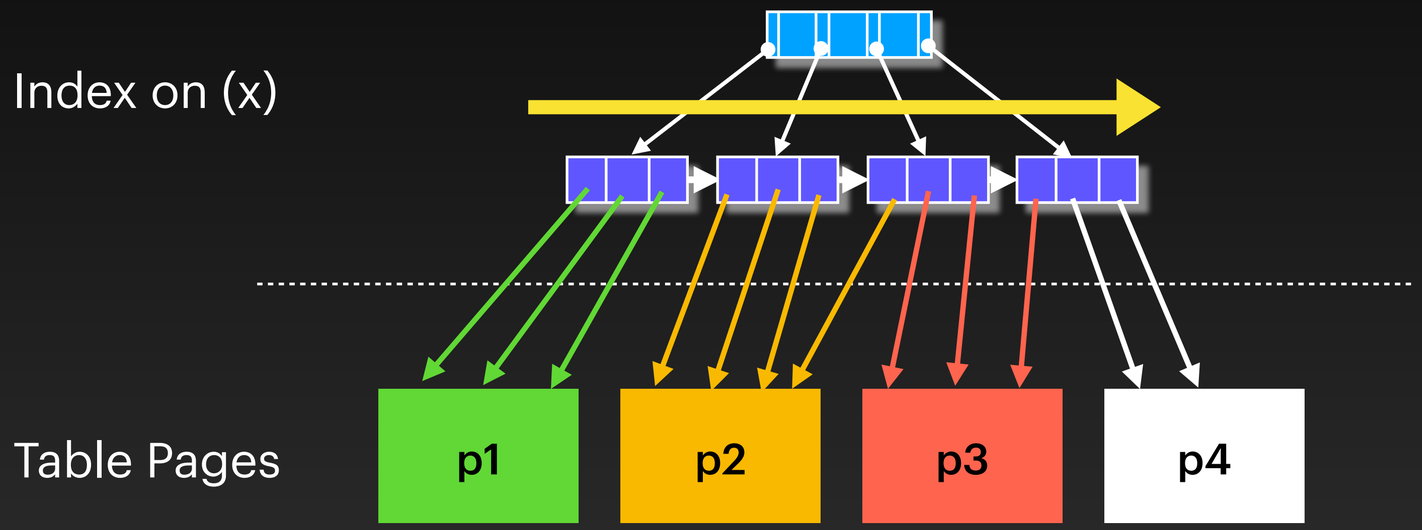
Non-clustered index



Data in index sorted but in random not sorted pages
lead to read multiple pages or the same page multiple time to get a range of data
Random Access to the disk which is slooooooooow.

Clustered index

update, sort the table is costly



sort data in pages and also in index.
no random access
seq access for the needed is better
tables are sorted only on one specific attribute so the table has just one clustered index BUT can have multiple non clustered indexes

why not set index on every attribute?
1. very large storage
2. cost on updates and insertion
Just use indexes on needed querirs.
ask urself about or, felxiblity of cols, updates on data.
Use desing advisor tools to help you choose the best indexes to build.