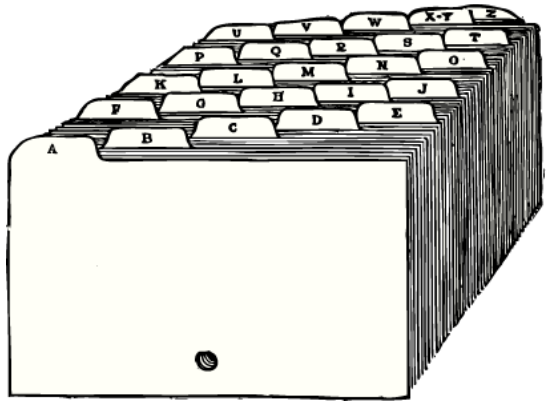
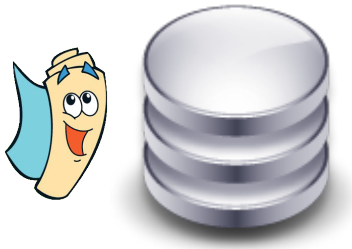
Conception Avancée de Bases de Données



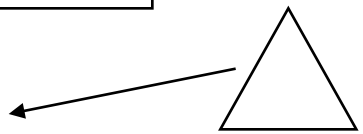
Index

Traduction en cours

From Ullman

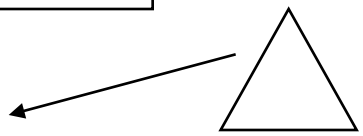


Parse Query



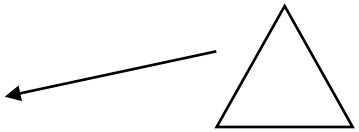
Query expression tree

Select Logical Query plan



Logical Query Plan tree

Select physical Query plan



Physical Query Plan tree

Relation Index

- Un index permet de localiser efficacement les tuples sans avoir à scanner l’ ensemble de la table
- La traversée d'un index fournit une liste d’ adresses de Tuples

- Types d’index
  - B+tree
  - Hash map
  - Bitmap
  - …

Heap Table Index

Index

Table

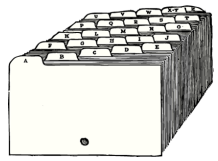
Key = “ **Bil** ”

Index
Aims
Alex
Barak
Ben
Betide
Bill
Bob
Curt
Jess
Jim
Kit
Mike
Nike
Rik
Ronny

7

1	Bob	1899
2	Mike	1212311
3	Jim	1312311
4	Alex	31231231
5	Curt	312311
6	Ronny	45564564
7	Bill	4556231
8	Ben	56454
9	Barak	89789471
10	Kit	2311889
11	Nike	48975
12	Jess	3216544
13	Rick	321648947
14	Aims	1218989
15	Betide	7489

Key



Heap Table Index : Content

Index

Table

Key = “ **Bil** ”

Aims
Alex
Barak
Ben
Betide
Bill
Bob
Curt
Jess
Jim
Kit
Mike
Nike
Rik
Ronny

7

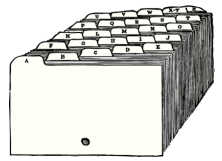
1	Bob	1899
2	Mike	1212311
3	Jim	1312311
4	Alex	31231231
5	Curt	312311
6	Ronny	45564564
7	Bill	4556231
8	Ben	56454
9	Barak	89789471
10	Kit	2311889
11	Nike	48975
12	Jess	3216544
13	Rick	321648947
14	Aims	1218989
15	Betide	7489

Key

INDEX

Value

HEAP TABLE



Heap Table Index building

Index

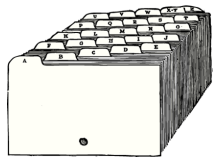
321648947	13
89789471	9
45564564	6
31231231	4
4556231	7
3216544	12
2311889	10
1312311	3
1218989	14
1212311	2
312311	5
56454	8
48975	11
7489	15
1899	1

Value

Table

1	Bob	1899
2	Mike	1212311
3	Jim	1312311
4	Alex	31231231
5	Curt	312311
6	Ronny	45564564
7	Bill	4556231
8	Ben	56454
9	Barak	89789471
10	Kit	2311889
11	Nike	48975
12	Jess	3216544
13	Rick	321648947
14	Aims	1218989
15	Betide	7489

Key



Sequential File index building

Index

Table

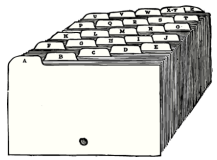
Key = “1212311”

321648947	13
89789471	9
45564564	6
31231231	4
4556231	7
3216544	12
2311889	10
1312311	3
1218989	14
1212311	2
312311	5
56454	8
48975	11
7489	15
1899	1

Value

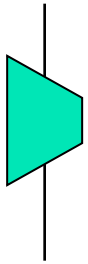
1	**Bob**	**1899**
2	**Mike**	**1212311**
3	**Jim**	**1312311**
4	**Alex**	**31231231**
5	**Curt**	**312311**
6	**Ronny**	**45564564**
7	**Bill**	**4556231**
8	**Ben**	**56454**
9	**Barak**	**89789471**
10	**Kit**	**2311889**
11	**Nike**	**48975**
12	**Jess**	**3216544**
13	**Rick**	**321648947**
14	**Aims**	**1218989**
15	**Betide**	**7489**

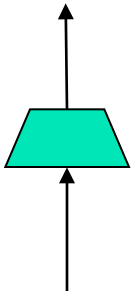
Key

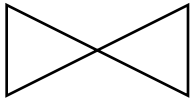


opérations

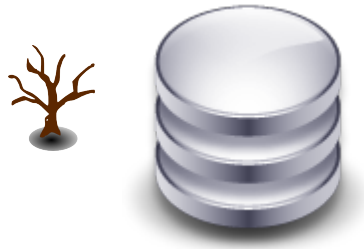
- Sélection
- Projection
- Jointure



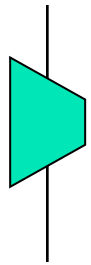




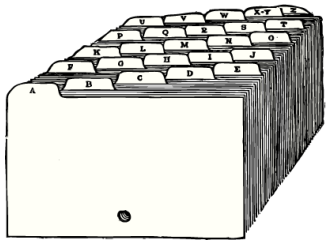
Sélection



- Recherche associative:
  - Accéder une donnée par son contenu et non par son adresse.
- Mise en oeuvre



  - Accès par adresse
  - Balayage séquentiel d'une table (seq scan)
  - Sélection par traversée d'index
  - Hash Table



Opérateurs Physiques

- Jointure :

SELECT * FROM R,S WHERE Y='E';

R

X	Y
5	A
8	B
3	B
9	E
1	G
10	J
7	K
2	U
4	V
6	Z

S

Y	Z
Z	20
B	22
E	28
K	24
M	25
N	30
U	27
L	23
V	21
X	26

Position du problème

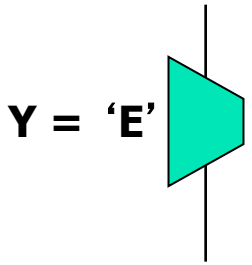
- Pour rechercher dans une table un Tuple ayant un attribut Y tel que Y = ‘E’

R

X	Y
5	A
8	B
3	B
9	E
1	G
10	J
7	K
2	U
4	V
6	Z

Y = ‘E’

9	E



Accès Physique à un Tuple : RID

Raw Identifier (RID)

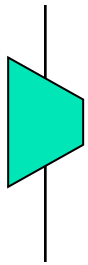
RID →

	X	Y
0	5	A
1	8	B
2	3	B
3	9	E
4	1	G
5	10	J
6	7	K
7	2	U
8	4	V
9	6	Z

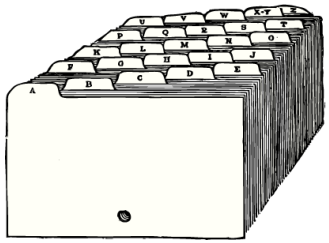
→ **3**

9	E

Y = 'E'



Complexité O (n)



Accès Physique à un Tuple : RID

Raw Identifier (RID)

Seq Scan

R

	X	Y
0	5	A
1	8	B
2	3	B
3	9	E
4	1	G
5	10	J
6	7	K
7	2	U
8	4	V
9	6	Z

RID

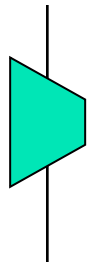
RID

Y = 'E'

3

9	E

R (3)



Seq Scan select : Y = ’E’

Seq Scan :

While i in index range

if Y = ‘E’ then OutPut = R(i)
Next (i);

End while

R

RID →

loop

RID →

	X	Y
0	5	A
1	8	B
2	3	B
3	9	E
4	1	G
5	10	J
6	7	K
7	2	U
8	4	V
9	6	Z

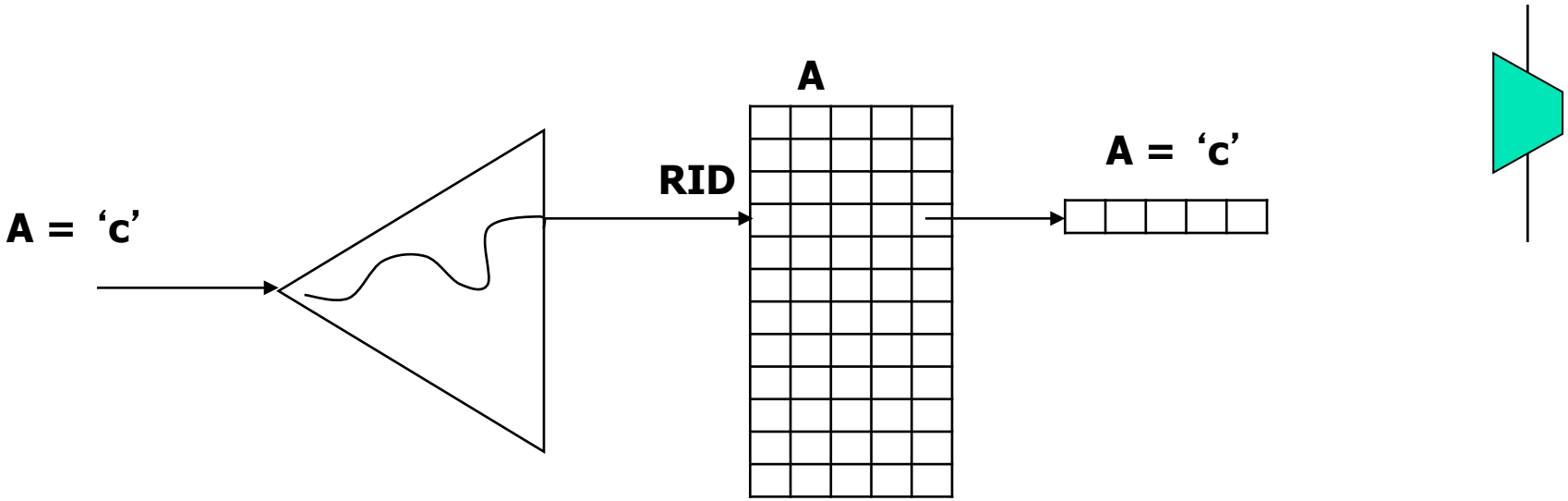
Complexité O (n)

Sélection par traversée d'index

- Un index permet de localiser efficacement les tuples sans avoir à scanner l'ensemble de la table
- La traversée d'un index fournit une liste d'adresses de Tuples

Sélection par traversée d'index

- La traversée d'un index fournit une liste d'adresses de Tuples

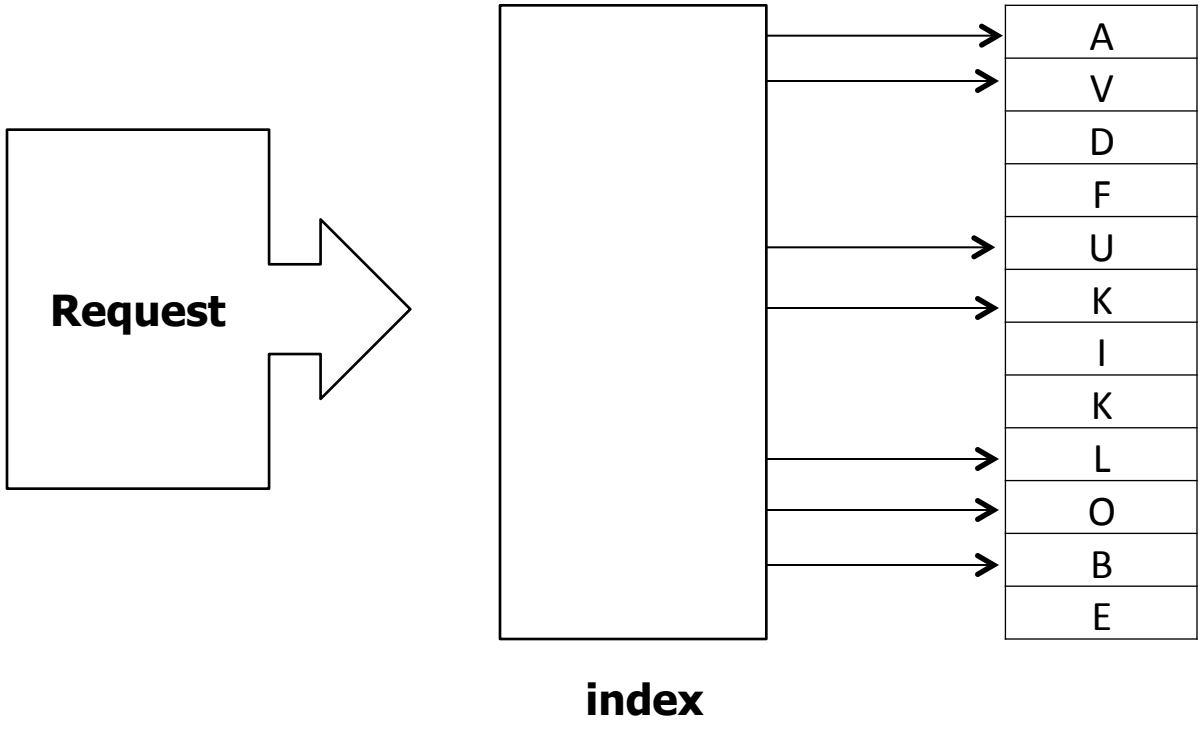


Types d'index

- B+tree
  - Arbre binaire sur disque
- Hash map
  - Table Key Value sur disque
    - Key : Valeur d'un Attribut
    - Value : Rid du Tuple
- Bit map
  - Table Key Value sur disque
    - Key : 0,1
    - Value : Rid du Tuple
- …

Index search VS Seq scan

- If a query is going to return a large portion of a table then the planner chooses a sequence scan over an index because it is actually faster



Indexes drawbacks

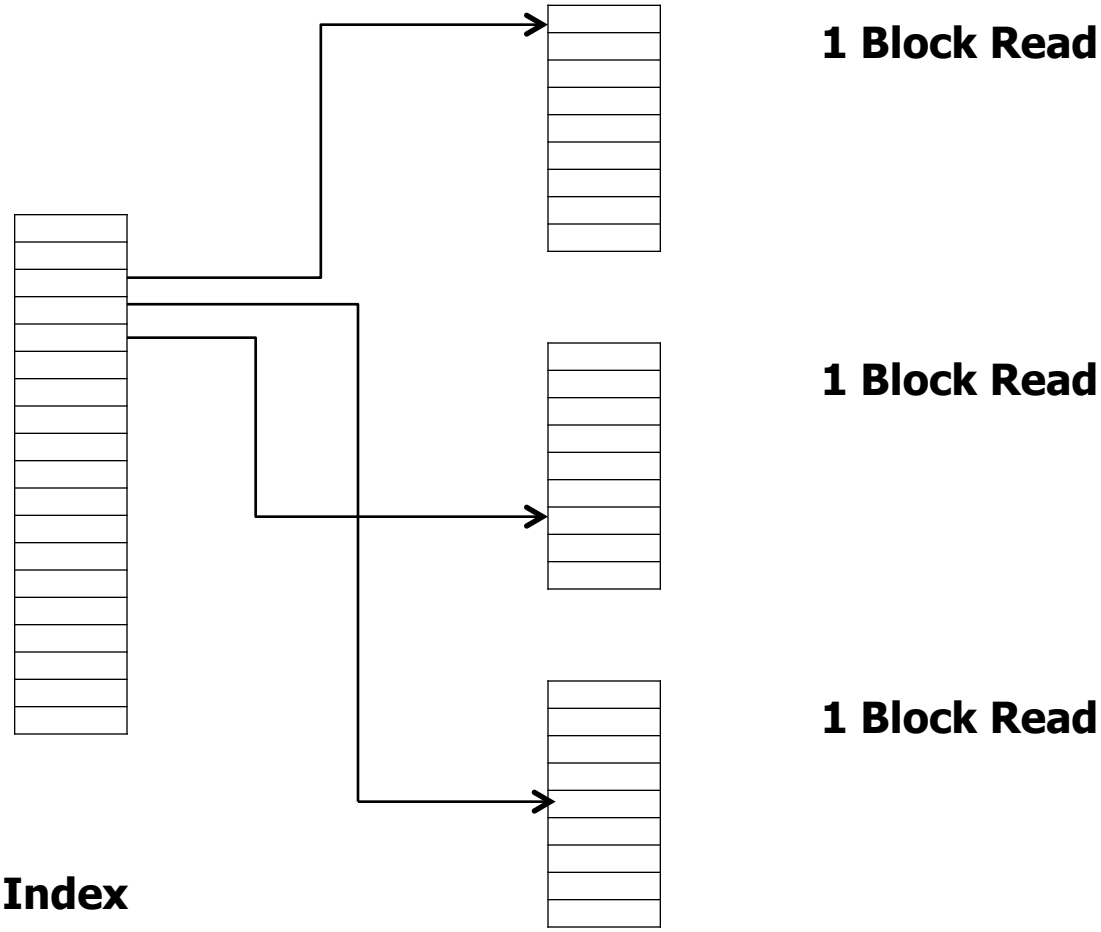
- Indexes are used only if the table is larger than a minimum size, and the query selects only a small percentage of the rows in the table.

- This is because the random disk access caused by an index scan can be slower than a straight read through the table, or sequential scan.

Indexes drawback

- Index scan requires several IO operations for each row.
- Whereas a sequential scan only requires a single IO for each row or even less because a disk block on the disk contains more than one row.
- Then more than one row can be fetched with a single IO operation.

Index and disk blocks



SQL WHERE condition

- If no SQL “WHERE” condition the DBMS Request Planner chooses a seq scan.
- If WHERE condition is not selective the Planner keep using seq scan
- If the WHERE condition is selective enough, the Request Planner may switch completely to an index scan.