Column-Store Tutorial

CSC443H1 Database System Technology

Niv Dayan

Question 1

Column A has N entries of which |A| are unique and |A| << N. It is subject to: “select B where x ≤ A and y ≥ A” where y > x. We want to compress the column and process the above query without decompressing the data.

(1) How do we achieve this when $1 < |A| < \log_2(N)$?

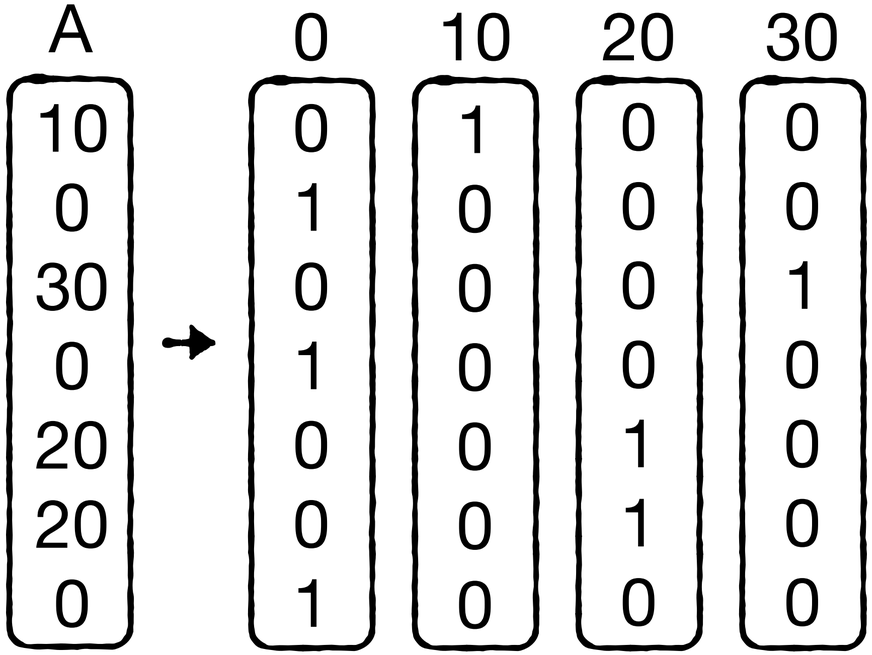
(2) How about when $\log_2(N) < |A| < \sqrt{N}$?

A
10
0
30
0
20
20
0

Question 1

(1) How do we achieve this when $1 < |A| < \log_2(N)$?

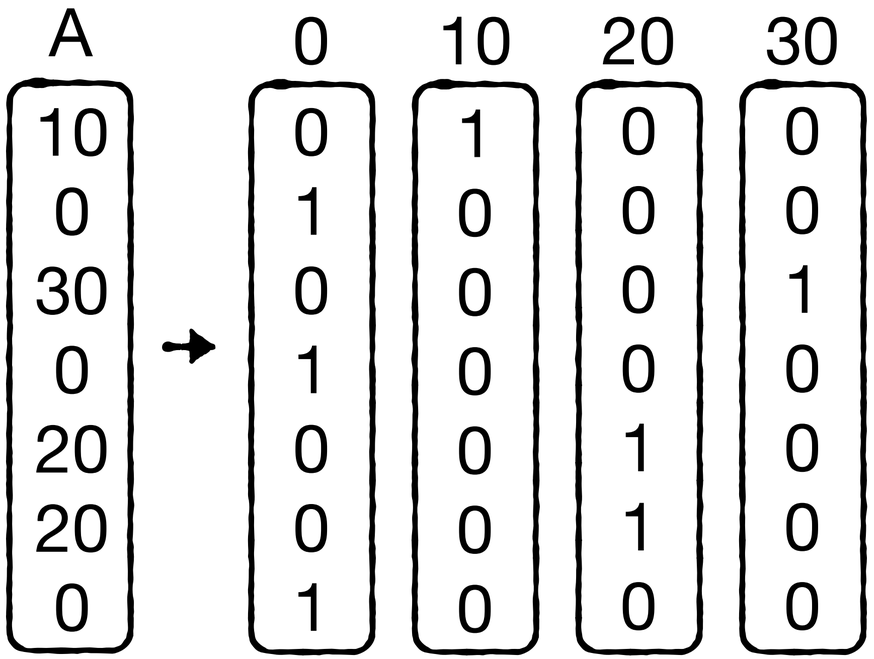
There are few unique values, so we can compress using bit-vector encoding



Question 1

(1) How do we achieve this when $1 < |A| < \log_2(N)$?

There are few unique values, so we can compress using bit-vector encoding



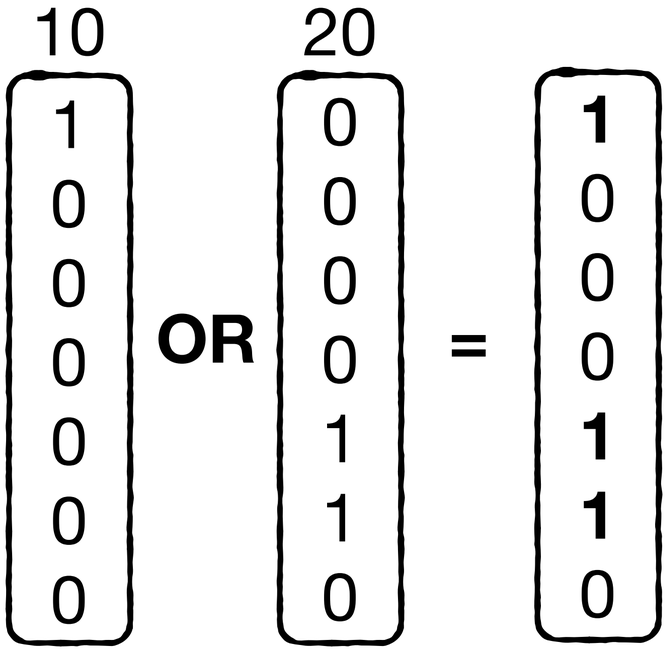
How to process queries?

e.g., select B where **10 ≤ A and 20 ≥ A**

Question 1

(1) How do we achieve this when $1 < |A| < \log_2(N)$?

There are few unique values, so we can compress using bit-vector encoding



How to process queries?

e.g., select B where 10 ≤ A and 20 ≥ A

“OR” the vectors of values in the range.

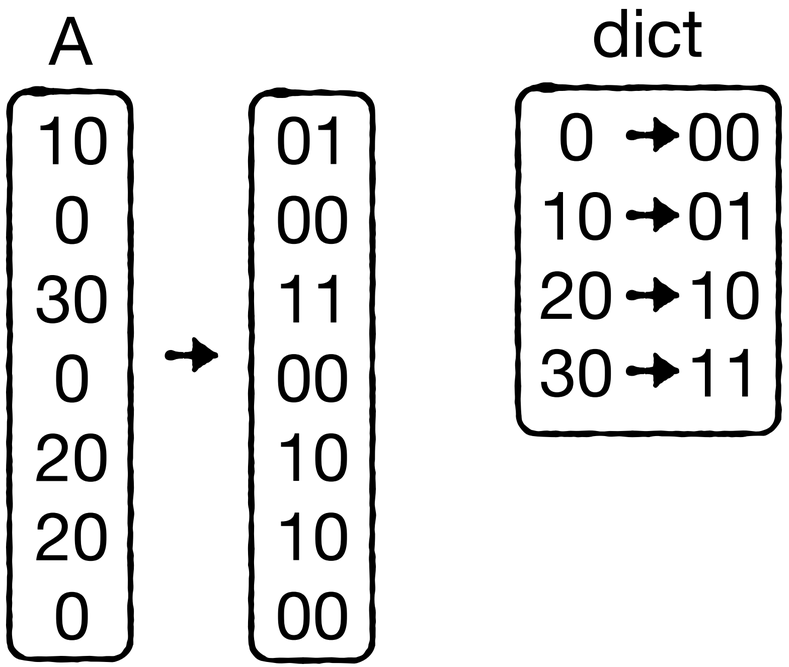
Question 1

(2) How about when $\log_2(N) < |A| < \sqrt{N}$?

Question 1

(2) How about when $\log_2(N) < |A| < \sqrt{N}$?

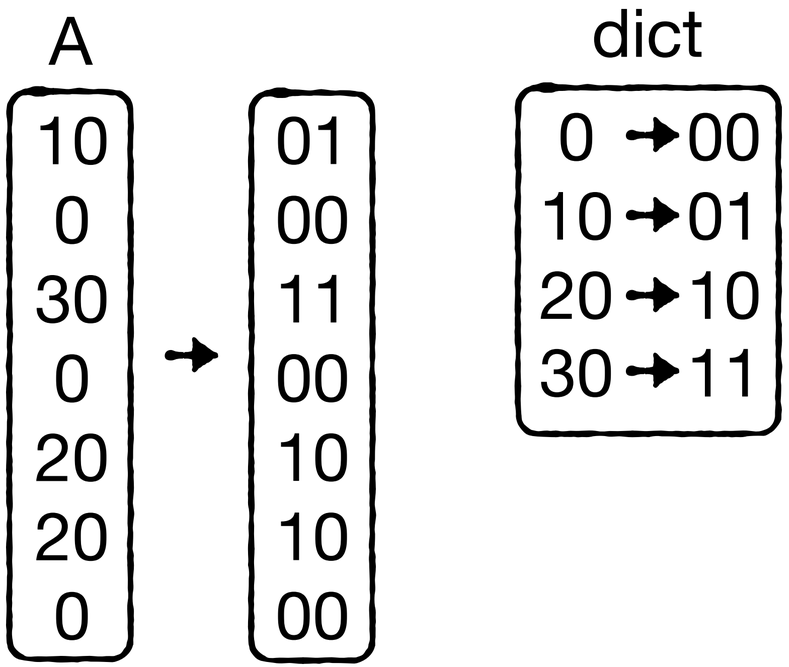
In this case, bit vector encoding may inflate rather than compress the data. Lets instead employ dictionary encoding.



Question 1

(2) How about when $\log_2(N) < |A| < \sqrt{N}$?

In this case, bit vector encoding may inflate rather than compress the data. Lets instead employ dictionary encoding.

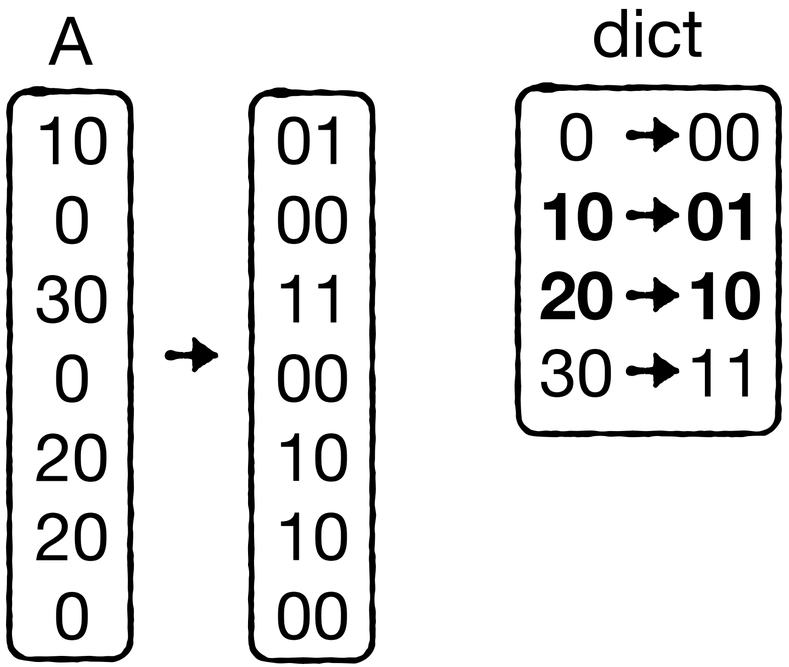


Note our dictionary is order-preserving.

Question 1

(2) How about when $\log_2(N) < |A| < \sqrt{N}$?

In this case, bit vector encoding may inflate rather than compress the data. Lets instead employ dictionary encoding.



Note our dictionary is order-preserving.

How to process queries?

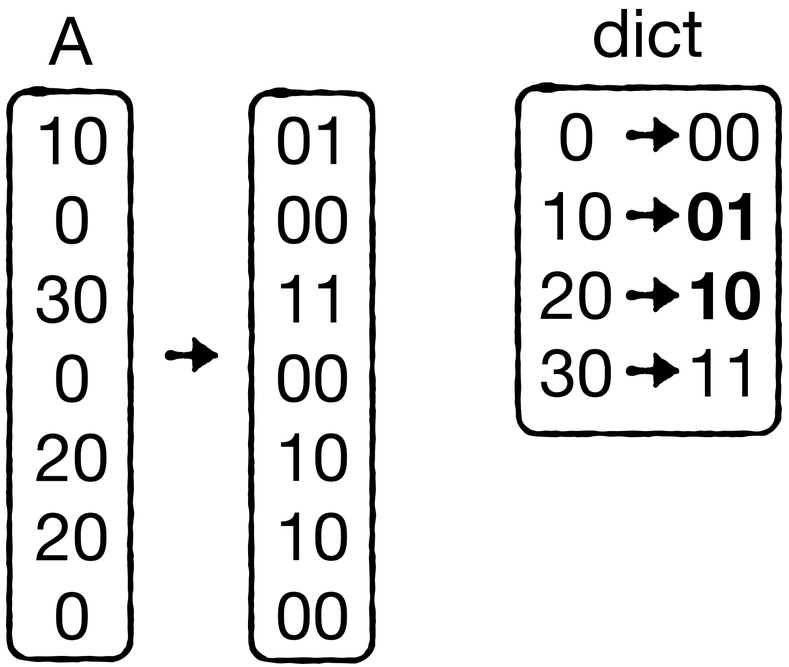
e.g., select B where **10 ≤ A** and **20 ≥ A**

Replace dictionary values in query

Question 1

(2) How about when $\log_2(N) < |A| < \sqrt{N}$?

In this case, bit vector encoding may inflate rather than compress the data. Lets instead employ dictionary encoding.



Note our dictionary is order-preserving.

How to process queries?

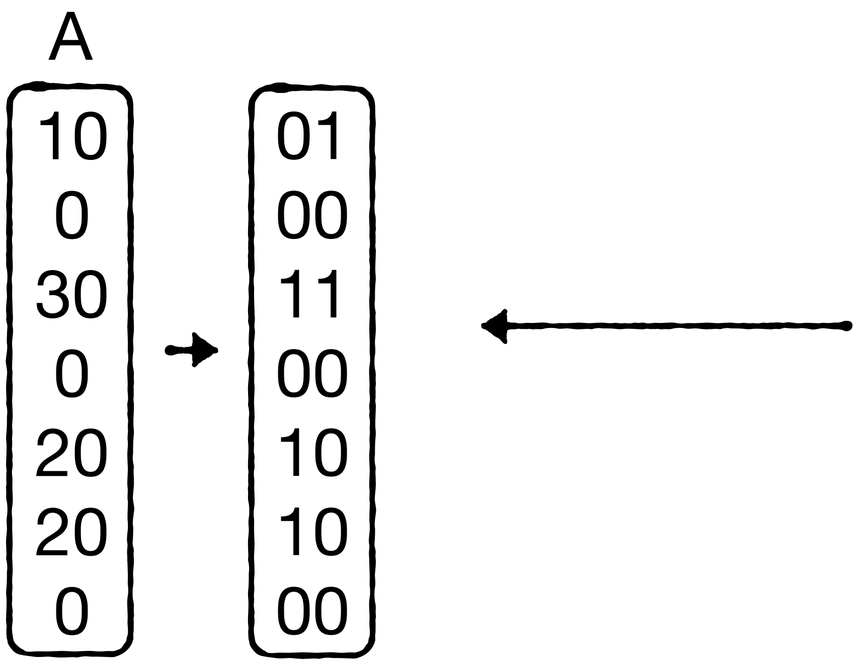
e.g., select B where **01 ≤ A** and **10 ≥ A**

Replace dictionary values in query

Question 1

(2) How about when $\log_2(N) < |A| < \sqrt{N}$?

In this case, bit vector encoding may inflate rather than compress the data. Lets instead employ dictionary encoding.



Note our dictionary is order-preserving.

How to process queries?

e.g., select B where **01 ≤ A** and **10 ≥ A**

Run on compressed column.

Question 2

Describe how to physically process the following query using late materialization. No information on cardinality is provided.

Relation R

A	B	C
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

Relation S

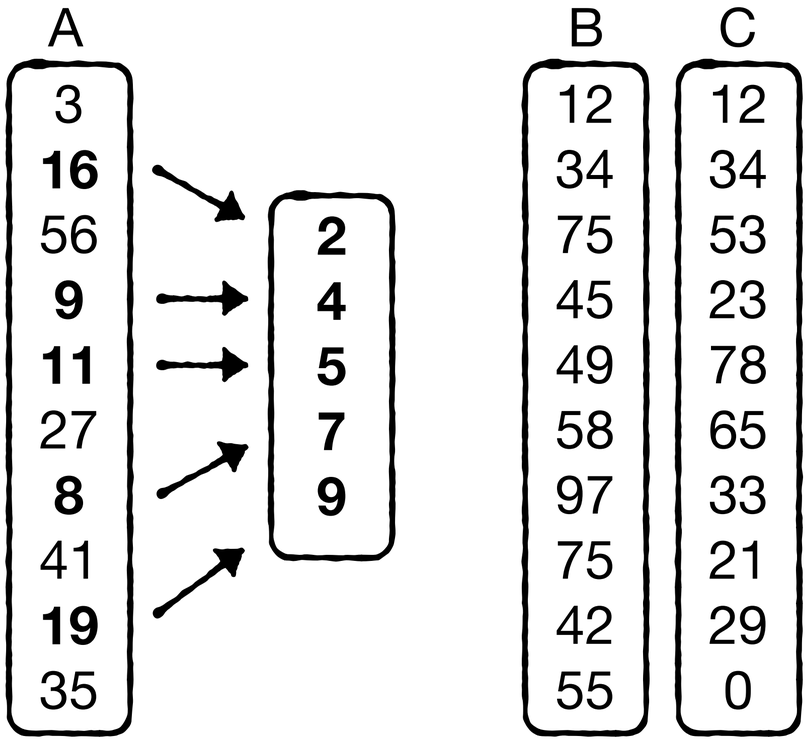
D	E
11	17
35	49
62	58
44	99
29	64
78	37
19	53
23	61
26	32
23	50

Select sum(A) from R, S where C=D and 5<A<20 and 40<B<50 and 55<E<65

Question 2

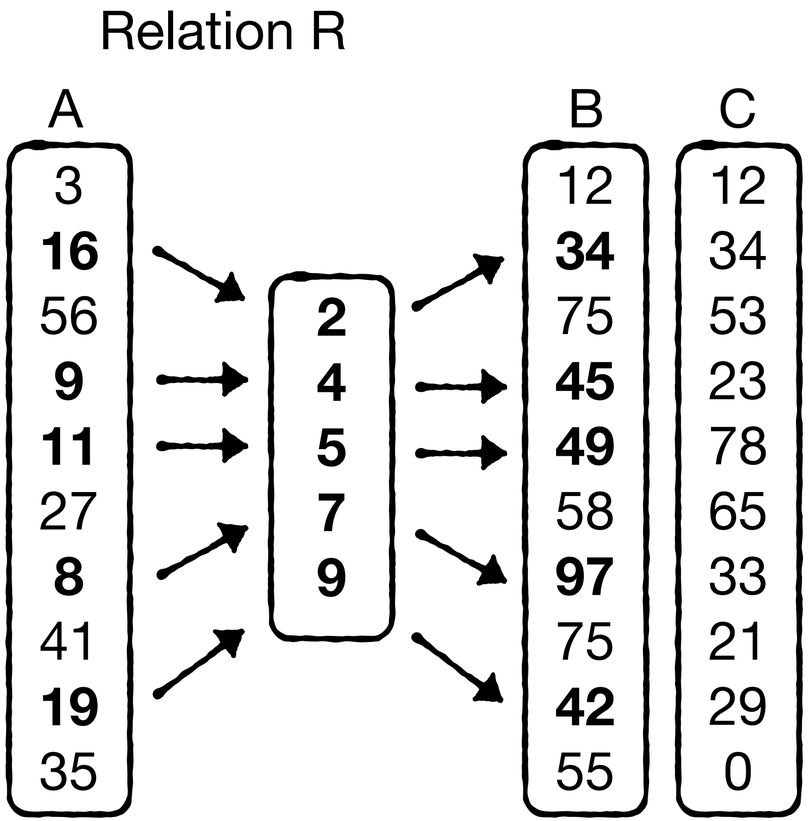
Select sum(A) from R, S where C=D and
5<A<20 and 40<B<50 and 55<E<65

Relation R



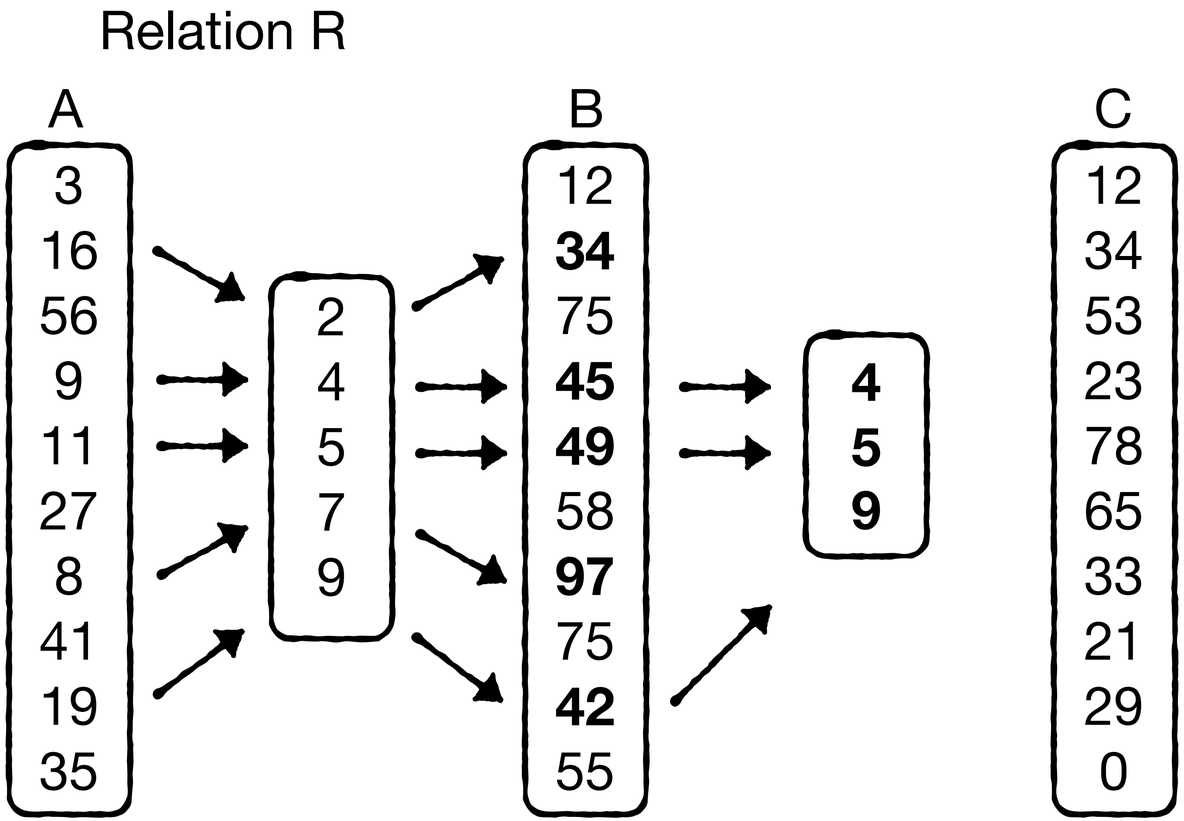
Question 2

Select sum(A) from R, S where C=D and
5<A<20 and 40<B<50 and 55<E<65



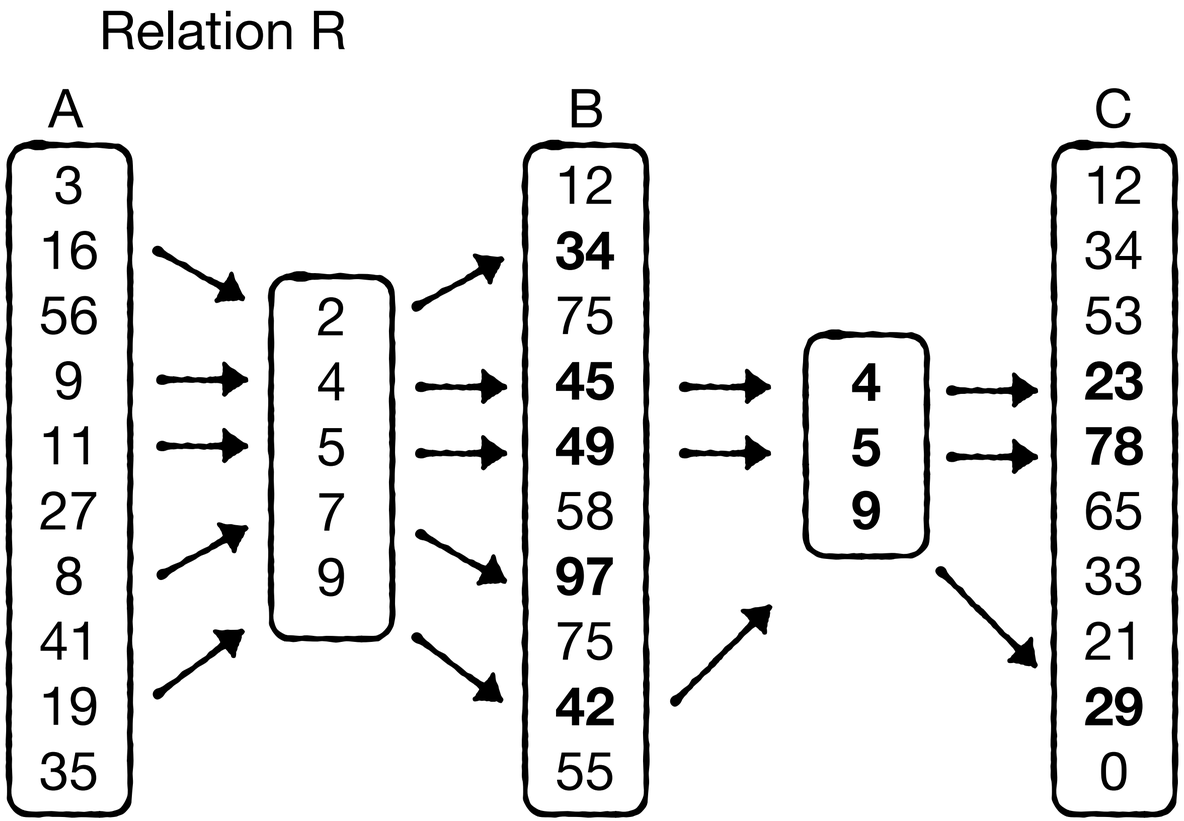
Question 2

Select sum(A) from R, S where C=D and
5<A<20 and **40<B<50** and 55<E<65



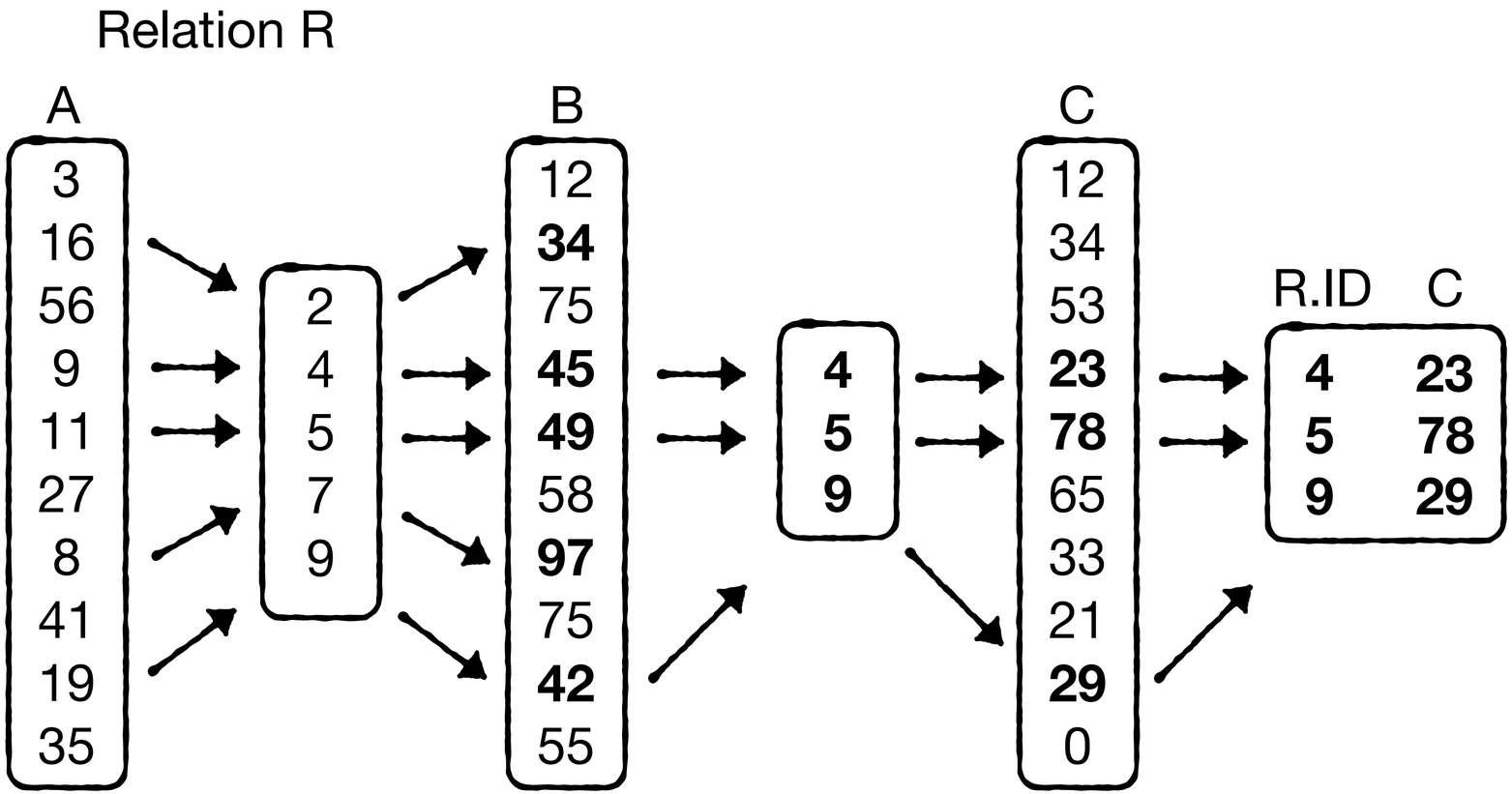
Question 2

Select sum(A) from R, S where C=D and
5<A<20 and **40<B<50** and 55<E<65



Question 2

Select sum(A) from R, S where C=D and
5<A<20 and **40<B<50** and 55<E<65



Question 2

Select sum(A) from R, S where C=D and
5<A<20 and **40<B<50** and 55<E<65

Relation R

A	B	C
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

R.ID	C
4	**23**
5	**78**
9	**29**

Relation S

D	E
11	17
35	49
62	58
44	99
29	64
78	37
19	53
23	61
26	32
23	50

Question 2

Select sum(A) from R, S where C=D and
5<A<20 and 40<B<50 and **55<E<65**

Relation R

A	B	C
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

R.ID	C
4	23
5	78
9	29

D
11
35
62
44
29
78
19
23
26
23

3
5
8

E
17
49
58
99
64
37
53
61
32
50

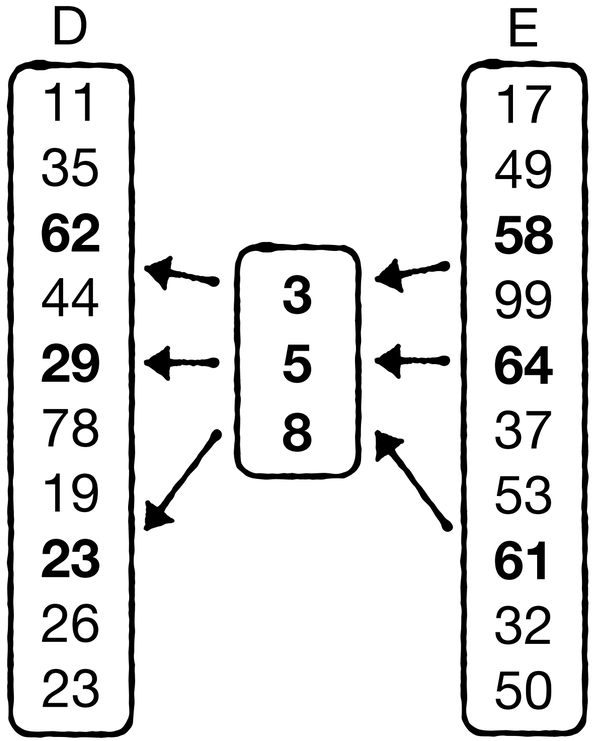
Question 2

Select sum(A) from R, S where C=D and
5<A<20 and 40<B<50 and **55<E<65**

Relation R

A	B	C
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

R.ID	C
4	23
5	78
9	29



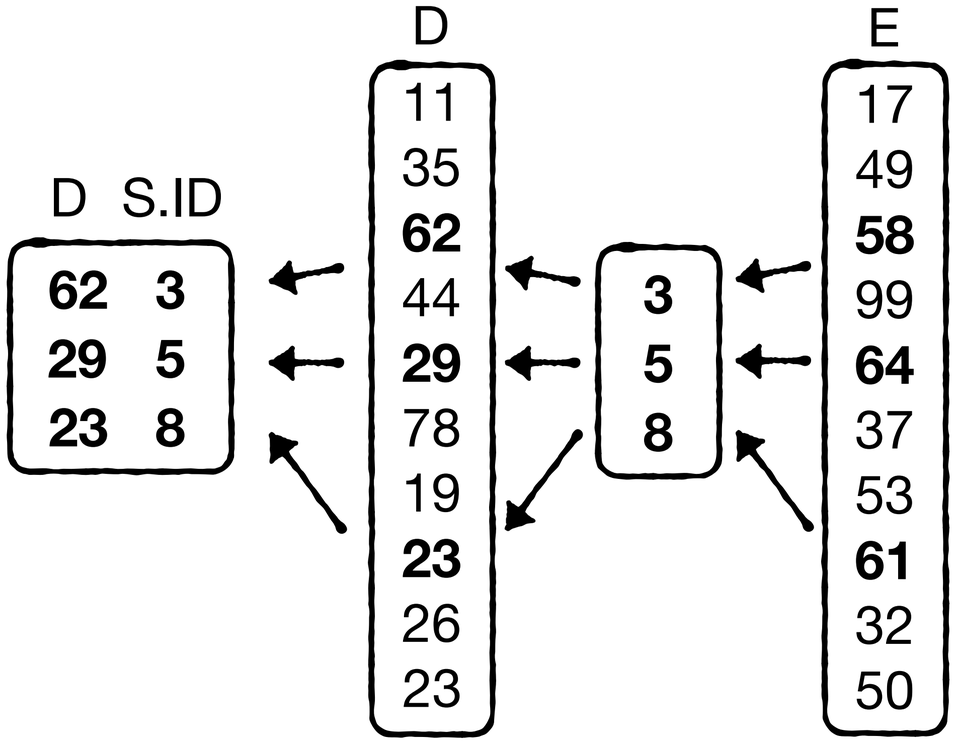
Question 2

Select sum(A) from R, S where C=D and
5<A<20 and 40<B<50 and **55<E<65**

Relation R

A	B	C
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

R.ID	C
4	23
5	78
9	29



Question 2

Select sum(A) from R, S where **C=D** and
5<A<20 and 40<B<50 and 55<E<65

Relation R

A	B	C
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0

R.ID	**C**
4	**23**
5	78
9	**29**

⋈

D	**S.ID**
62	3
29	**5**
23	**8**

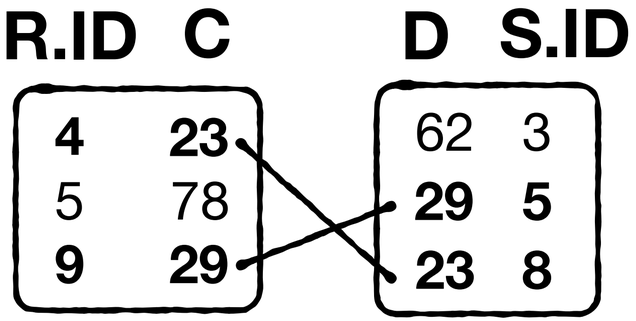
D	E
11	17
35	49
62	58
44	99
29	64
78	37
19	53
23	61
26	32
23	50

Question 2

Select sum(A) from R, S where **C=D** and
5<A<20 and 40<B<50 and 55<E<65

Relation R

A	B	C
3	12	12
16	34	34
56	75	53
9	45	23
11	49	78
27	58	65
8	97	33
41	75	21
19	42	29
35	55	0



R.ID	S.ID
4	**8**
9	**5**

D	E
11	17
35	49
62	58
44	99
29	64
78	37
19	53
23	61
26	32
23	50

Question 2

Select **sum(A)** from R, S where C=D and
5<A<20 and 40<B<50 and 55<E<65

Relation R

A
3
16
56
9
11
27
8
41
19
35

R.ID	S.ID
4	8
9	5

Question 2

Select **sum(A)** from R, S where C=D and
5<A<20 and 40<B<50 and 55<E<65

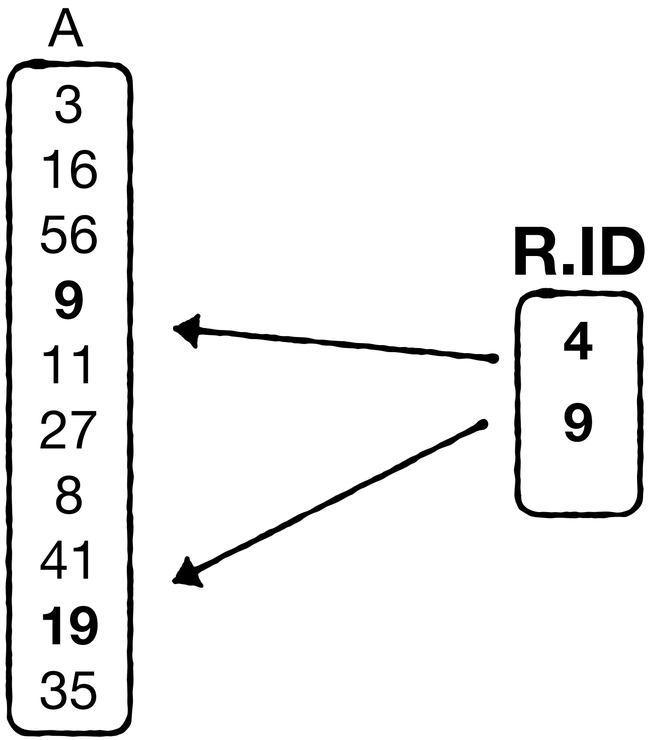
Relation R

A
3
16
56
9
11
27
8
41
19
35

R.ID
4
9

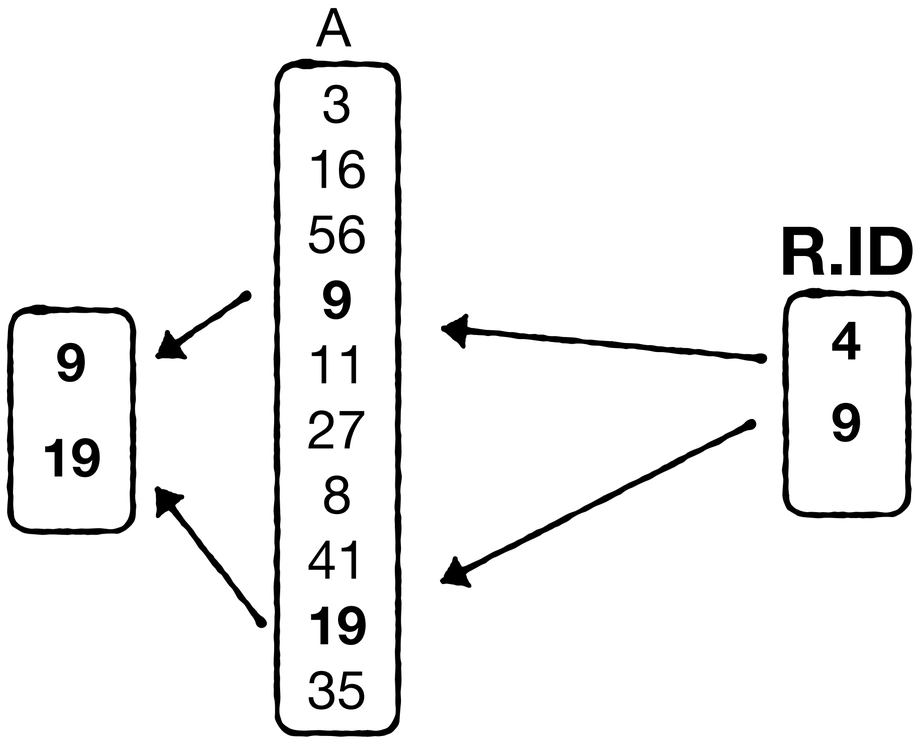
Question 2

Select **sum(A)** from R, S where C=D and
5<A<20 and 40<B<50 and 55<E<65



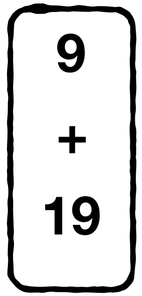
Question 2

Select **sum(A)** from R, S where C=D and
5<A<20 and 40<B<50 and 55<E<65



Question 2

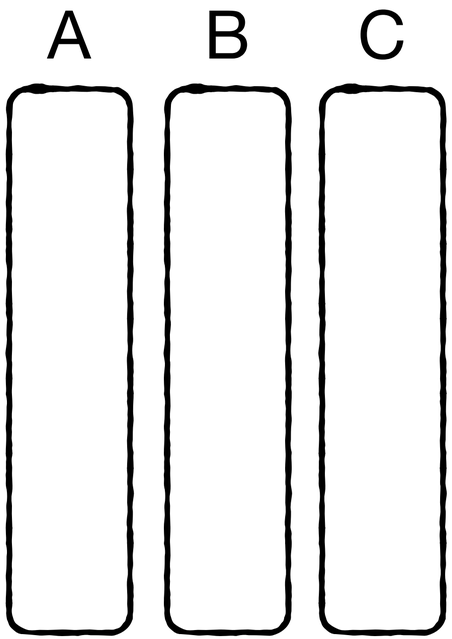
Select **sum(A)** from R, S where C=D and
5<A<20 and 40<B<50 and 55<E<65



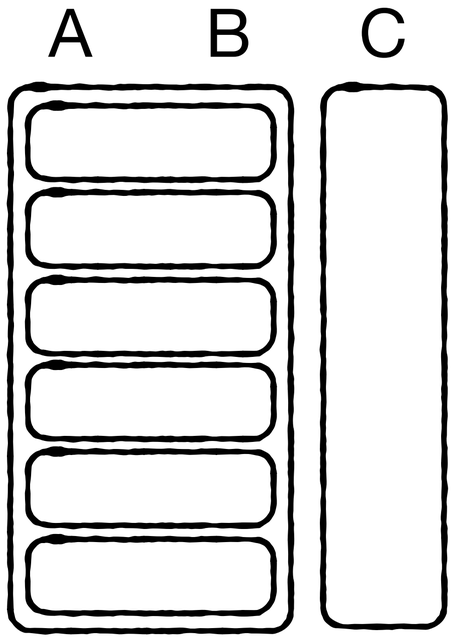
= 28

Question 3

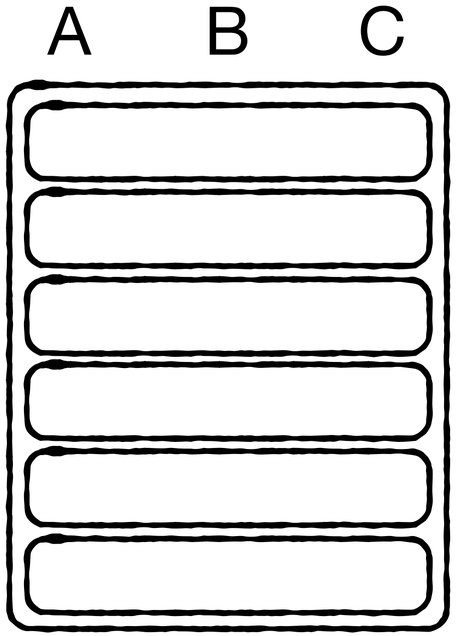
“Column groupings” are a way to store values from across several columns in a table in a row-store format. In which cases is it beneficial to group columns, and in which cases is it harmful?



Example 1



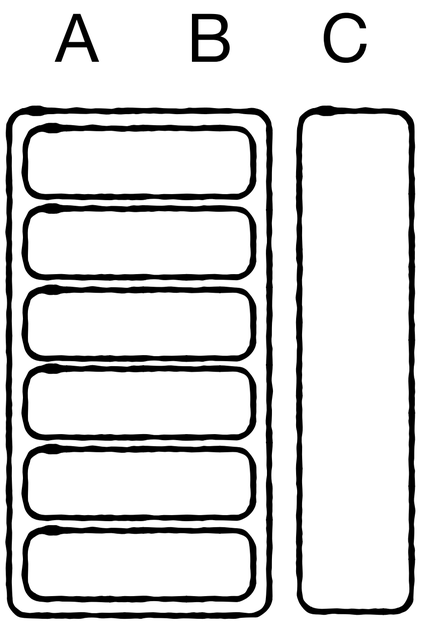
Example 2



Example 3

Question 3

“Column groupings” are a way to store values from across several columns in a table in a row-store format. In which cases is it beneficial to group columns, and in which cases is it harmful?



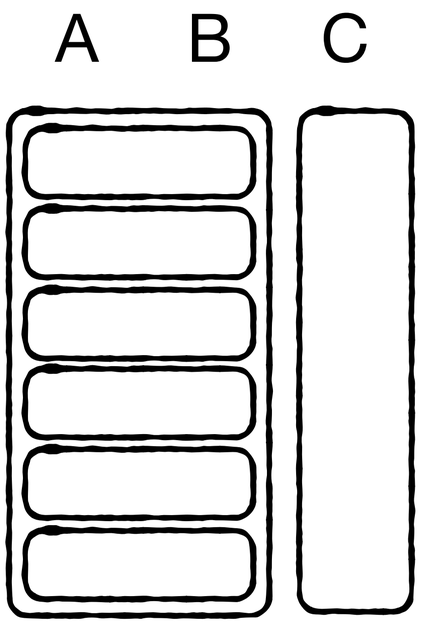
Good when values of corresponding rows along grouped columns are always accessed at the same time

Example:

Select A, B, C where C=“”

Question 3

“Column groupings” are a way to store values from across several columns in a table in a row-store format. In which cases is it beneficial to group columns, and in which cases is it harmful?



It’s harmful for queries that access only a subset of entries in a group, or that don’t require some column in a group

Example:

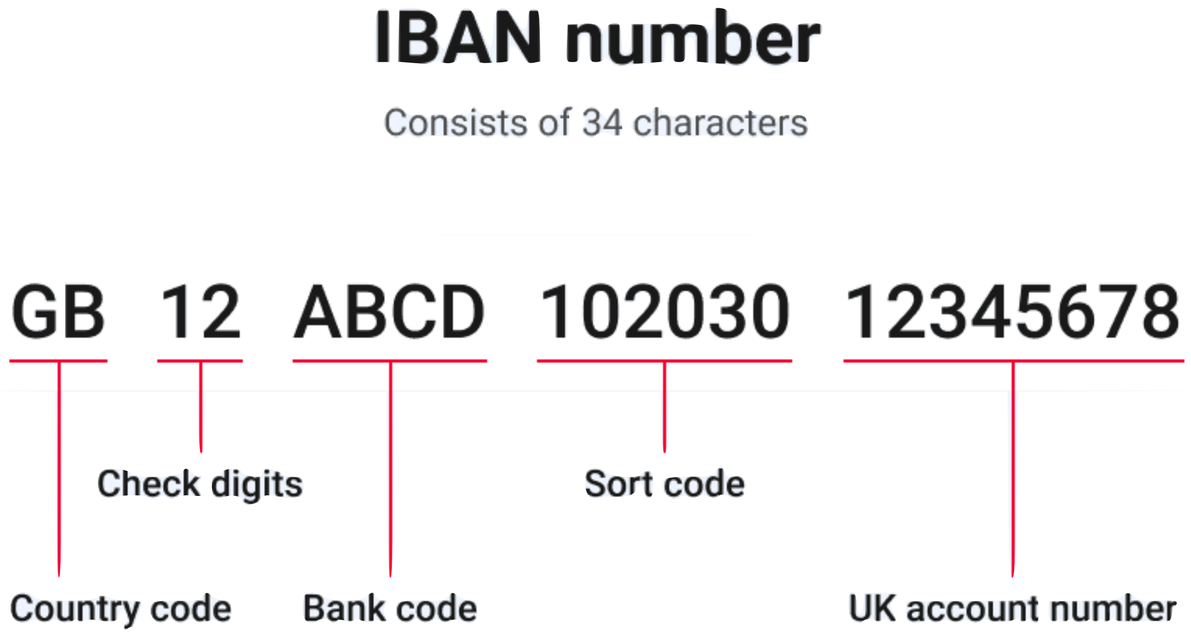
Select A, C

Select B, C where B=“”

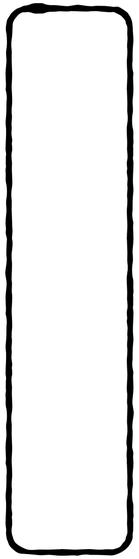
Select A, B where A = “…”

Question 4

Consider a column of International Bank Account Numbers (IBANs). We have queries of the form “select * where IBAN = x”. How can we optimize for such queries without sorting the column or using an index? Propose a generic solution that’s suitable beyond just IBANs.



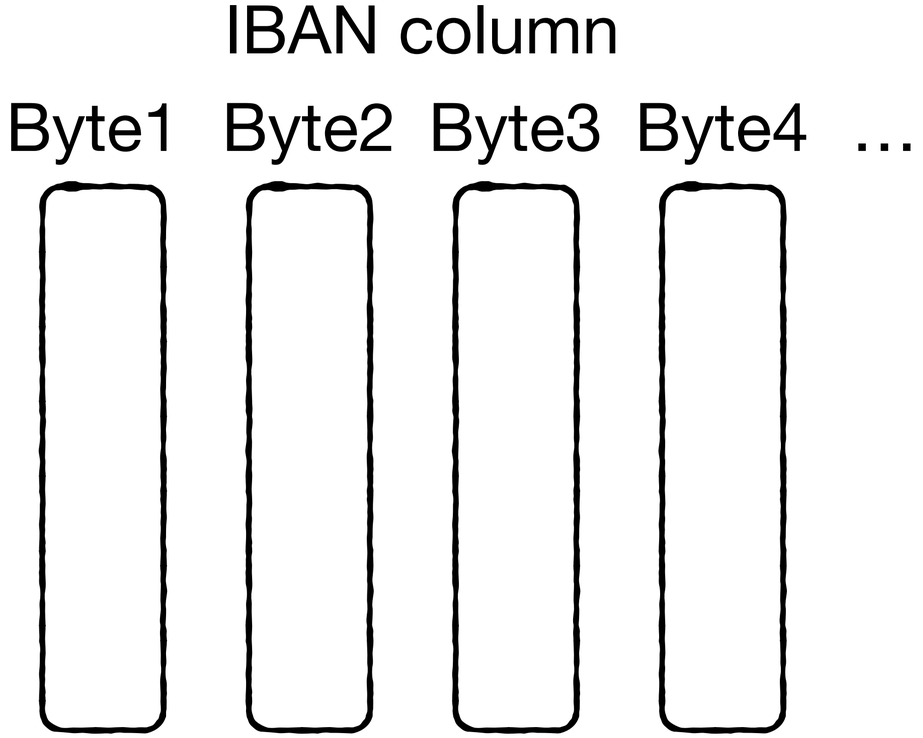
IBAN column



Question 4

Consider a column of International Bank Account Numbers (IBANs). We have queries of the form “select * where IBAN = x”. How can we optimize for such queries without sorting the column or using an index? Propose a generic solution that’s suitable beyond just IBANs.

Byte weaving: store each byte as a column.



Question 4

Consider a column of International Bank Account Numbers (IBANs). We have queries of the form “select * where IBAN = x”. How can we optimize for such queries without sorting the column or using an index? Propose a generic solution that’s suitable beyond just IBANs.

Byte weaving: store each byte as a column.

Traverse columns from most selective byte first and employ late materialization.