PASS
Data Community
SUMMIT
2022

SQL Server Partitioning from Zero to Hero

Daniel Hutmacher

(he/him)

Principal Consultant

Structured Concepts

Daniel Hutmacher

Principal consultant

Structured Concepts

@dhmacher

sqlsunday.com

strd.co

I'm a SQL Server consultant with over 20 years of experience.

Data Platform MVP, conference organizer, and blogger.

Big fan of travel, coffee, and dogs.

I live with my wonderful wife in Stockholm, Sweden.

In this session:

Thank you

Daniel Hutmacher

daniel@strd.co

@dhmacher

sqlsunday.com

PASS
Data Community
SUMMIT
2022

Session evaluation

Your feedback is important to us

Evaluate this session at:

www.PASSDataCommunitySummit.com/evaluation

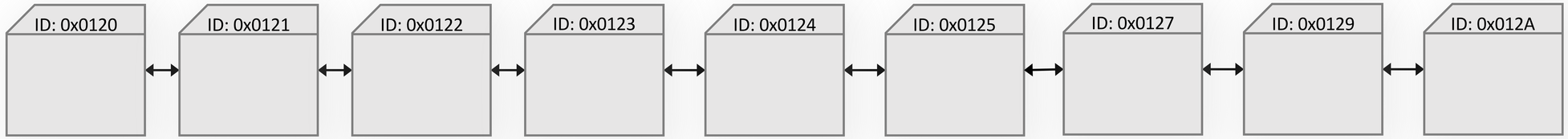
PASS
Data Community
SUMMIT
2022

In this session:

- Some queries can be tuned to use a single partition (partition elimination)
- Can solve specific latch contention issues by spreading writes
- You can backup only read-write partitions –> faster backups.
- You can restore the important partitions first –> shorter RTO.

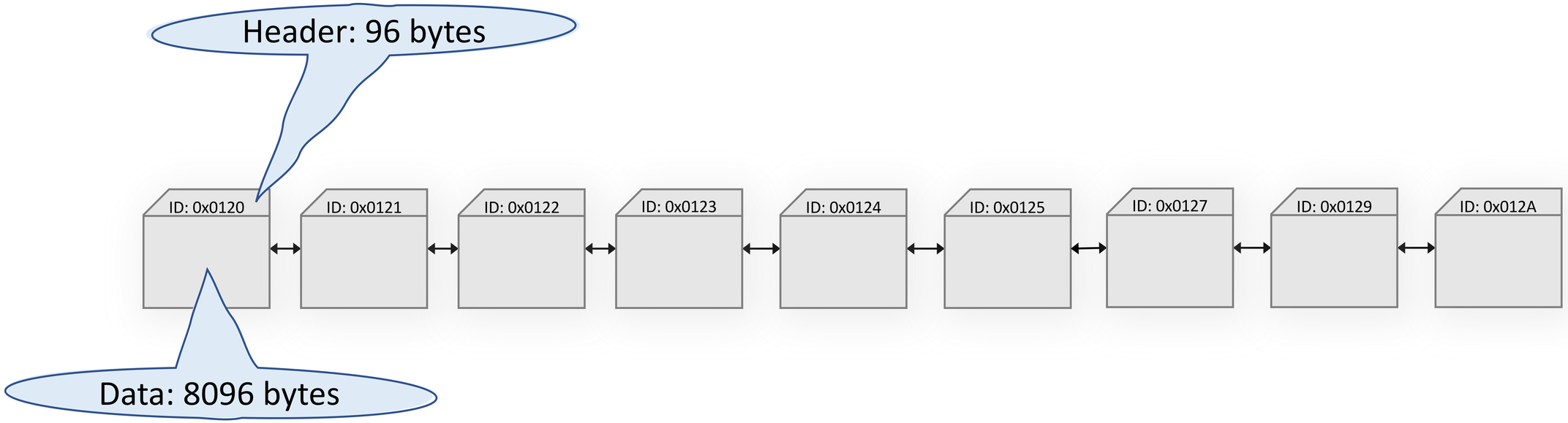
Simplifications ahead

How pages work



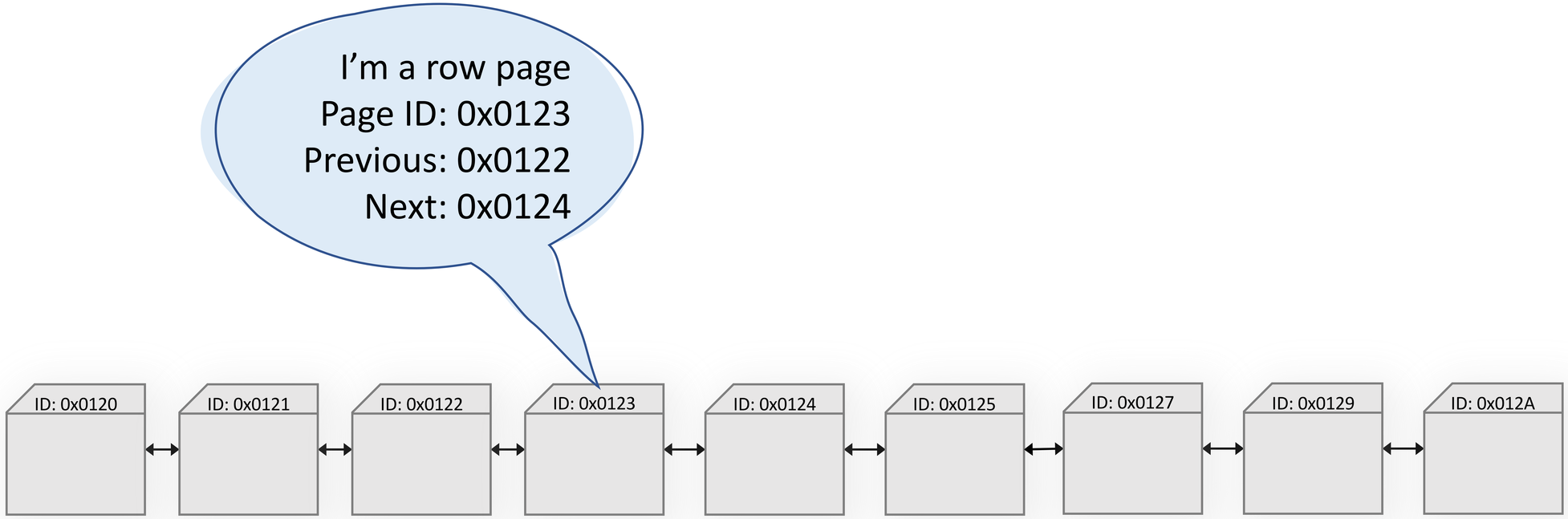
PASS
Data Community
SUMMIT
2022

How pages work



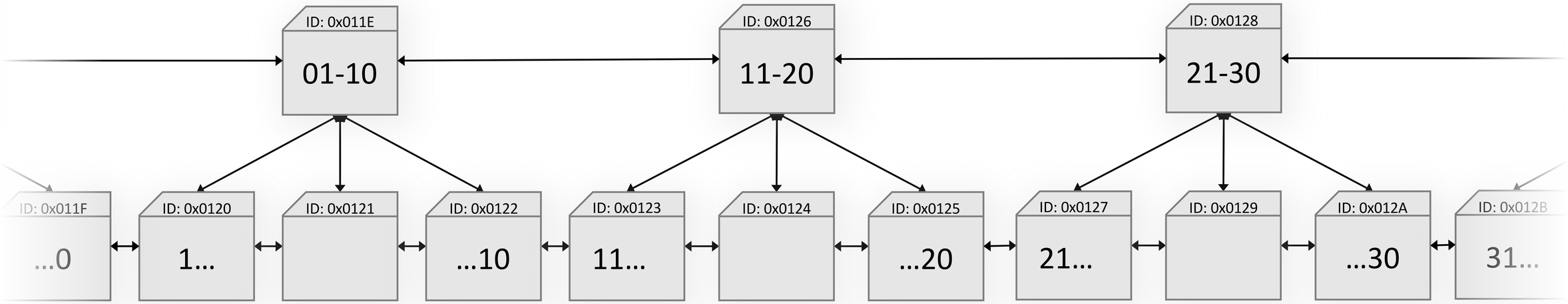
PASS
Data Community
SUMMIT
2022

How pages work



PASS Data Community SUMMIT 2022

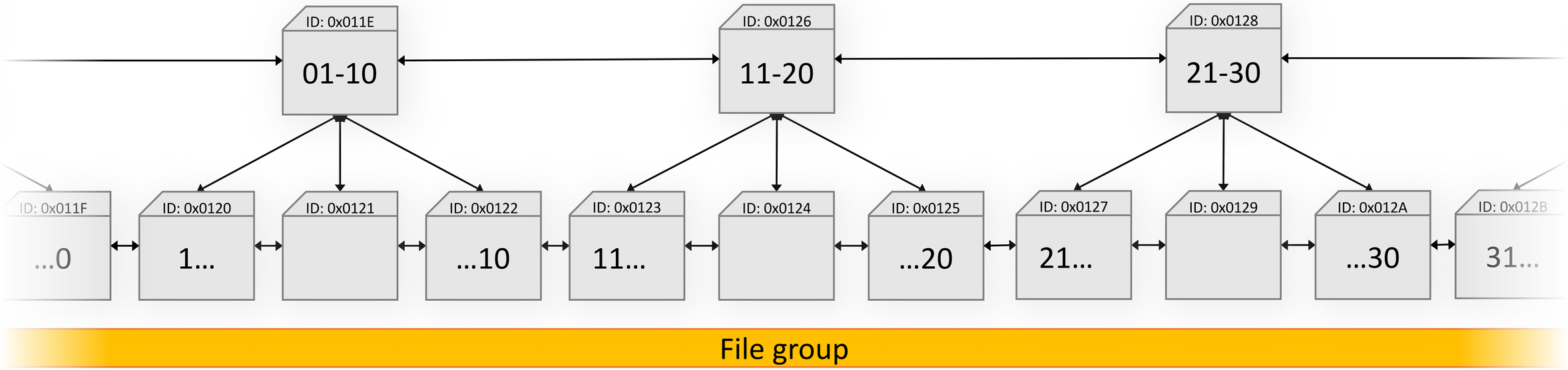
Indexes can have multiple levels



PASS Data Community SUMMIT 2022

What is partitioning?

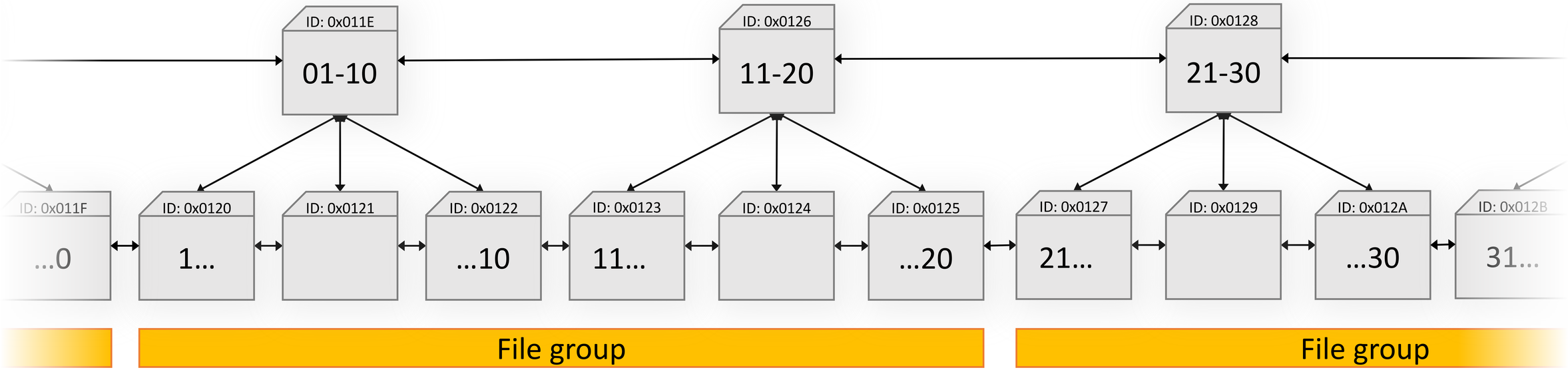
Regular table: one partition on one filegroup



PASS Data Community SUMMIT 2022

What is partitioning?

Partitioned table: *multiple* partitions, on *one or more* filegroups.



PASS Data Community SUMMIT 2022

Setting up partitioning

How: the partition function

```
CREATE PARTITION FUNCTION AnnualFunction(date)
AS RANGE RIGHT
        FOR VALUES ( '2020-01-01' , '2021-01-01' , '2022-01-01' );
```

PASS Data Community SUMMIT 2022

Setting up partitioning

How: the partition function

```
CREATE PARTITION FUNCTION AnnualFunction(date)
AS RANGE RIGHT
        FOR VALUES ( '2020-01-01' , '2021-01-01' , '2022-01-01' );
```

2020-01-01 – 2020-12-31 2021-01-01 – 2021-12-31 2022-01-01 – 2022-12-31

PASS Data Community SUMMIT 2022

Setting up partitioning

```
CREATE PARTITION FUNCTION AnnualFunction(date)
AS RANGE RIGHT
        FOR VALUES ( '2020-01-01' , '2021-01-01' , '2022-01-01' );

CREATE PARTITION SCHEME Annual
AS PARTITION AnnualFunction
TO ( [FILEGROUP_A] , [FILEGROUP_B] , [FILEGROUP_C] , [FILEGROUP_D] );
```

Where: the partition scheme

PASS Data Community SUMMIT 2022

Setting up partitioning

```
CREATE PARTITION FUNCTION AnnualFunction(date)
AS RANGE RIGHT
       FOR VALUES ( '2020-01-01' , '2021-01-01' , '2022-01-01' );
```

```
CREATE PARTITION SCHEME Annual
AS PARTITION AnnualFunction
TO ( [FILEGROUP_A] , [FILEGROUP_B] , [FILEGROUP_C] , [FILEGROUP_D] );
```

```
CREATE TABLE dbo.partitioned_table (
    TransactionDate    date NOT NULL,
    ...
    ...
    ...
) ON Annual(TransactionDate);
```

What: assigning the index to the partition scheme

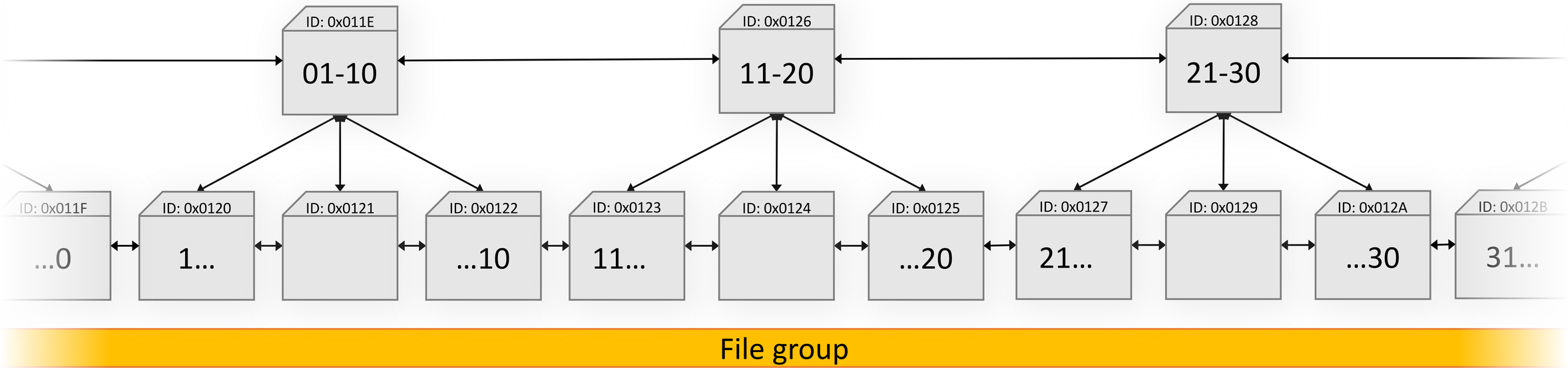
PASS Data Community SUMMIT 2022

DEMO

Image credit: Eric Zhang, Unsplash

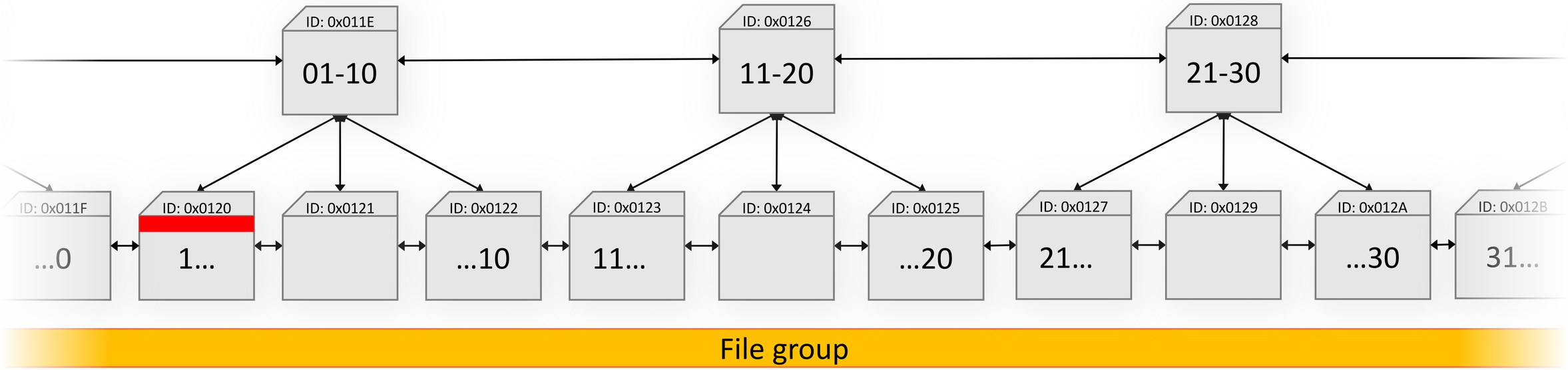
A regular delete

```
DELETE FROM dbo.regular_table
WHERE id BETWEEN 1 AND 20;
```



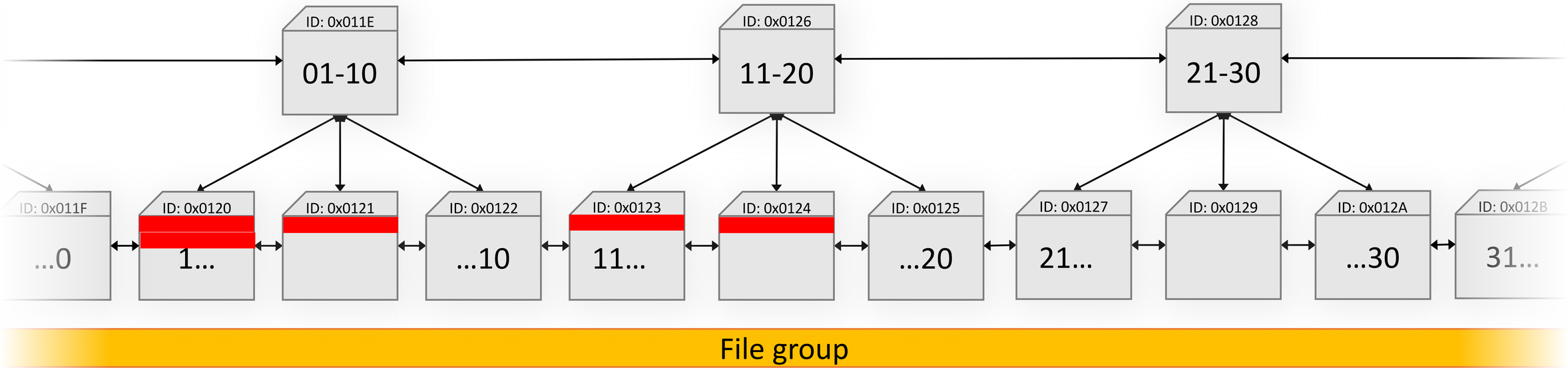
PASS Data Community SUMMIT 2022

A regular delete



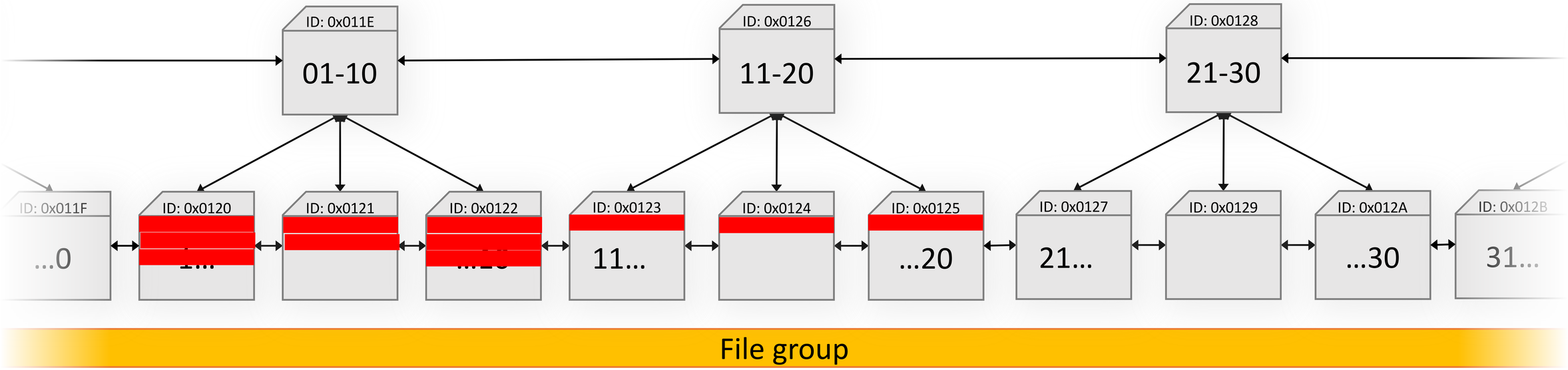
PASS Data Community SUMMIT 2022

A regular delete



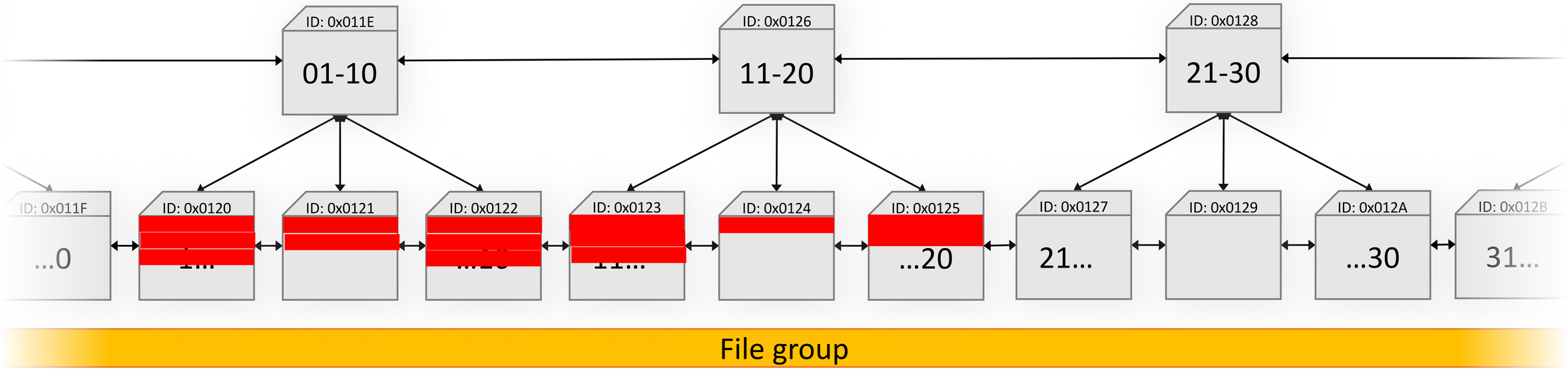
PASS Data Community SUMMIT 2022

A regular delete



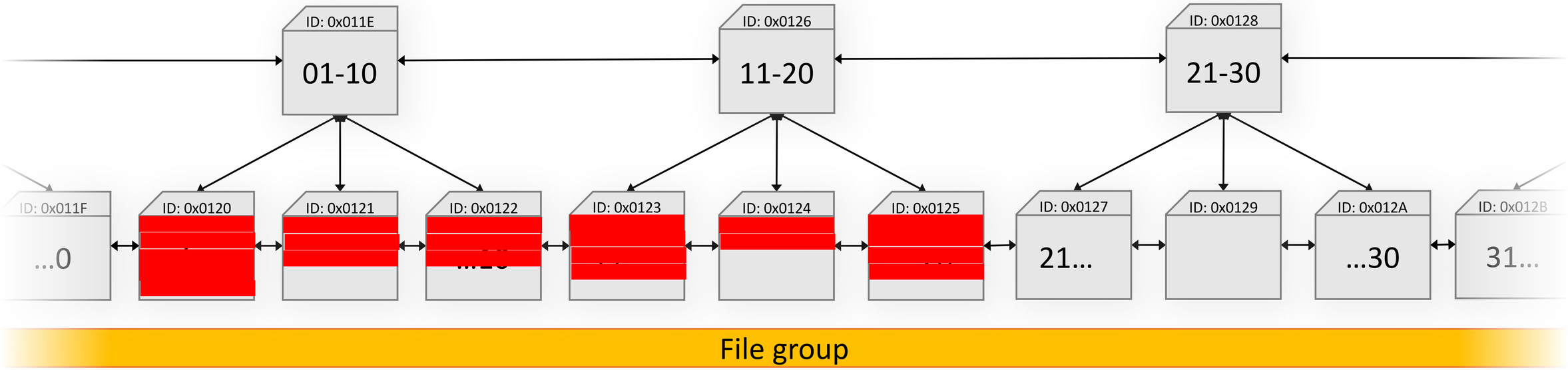
PASS Data Community SUMMIT 2022

A regular delete



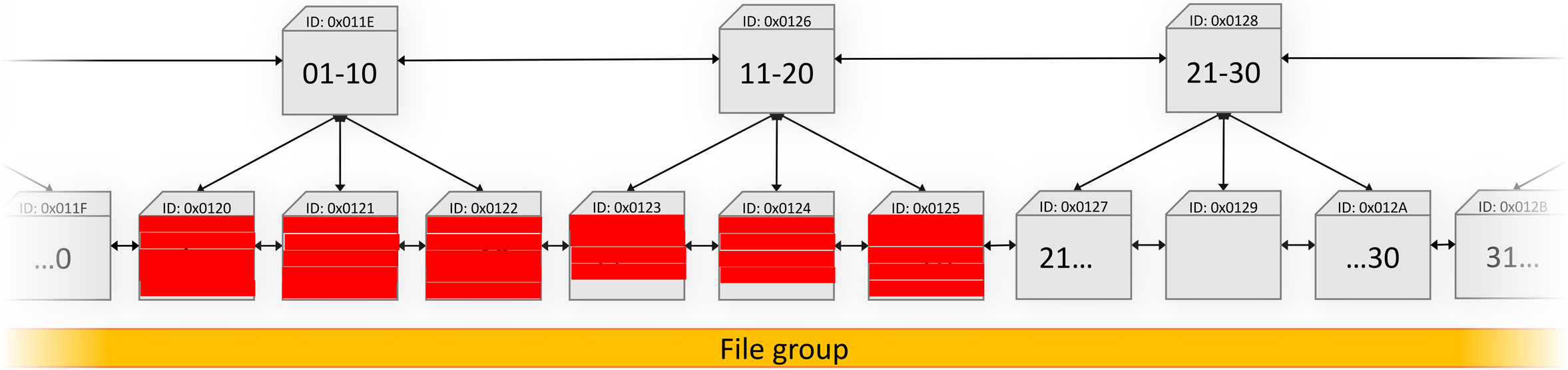
PASS Data Community SUMMIT 2022

A regular delete



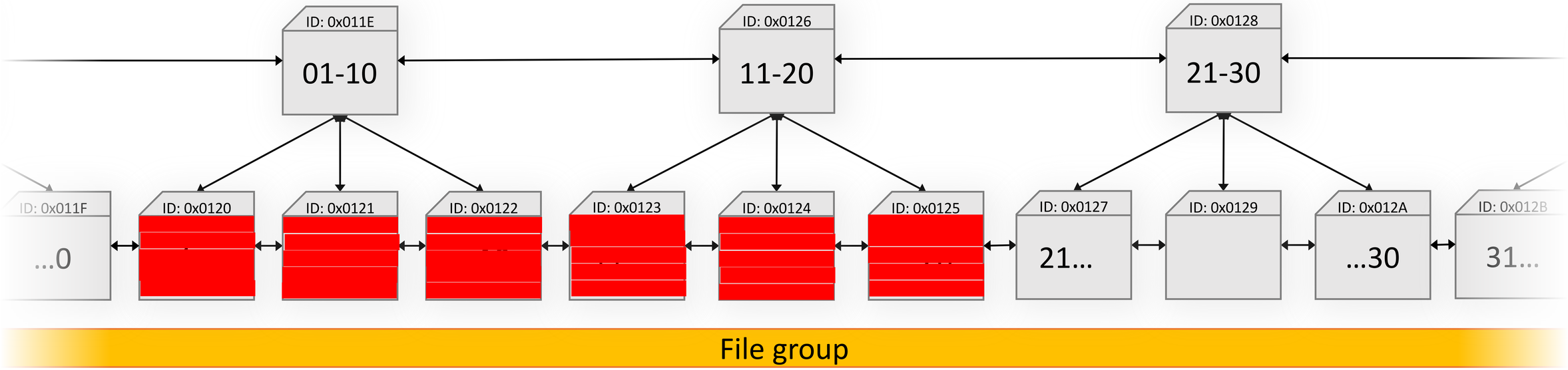
PASS Data Community SUMMIT 2022

A regular delete



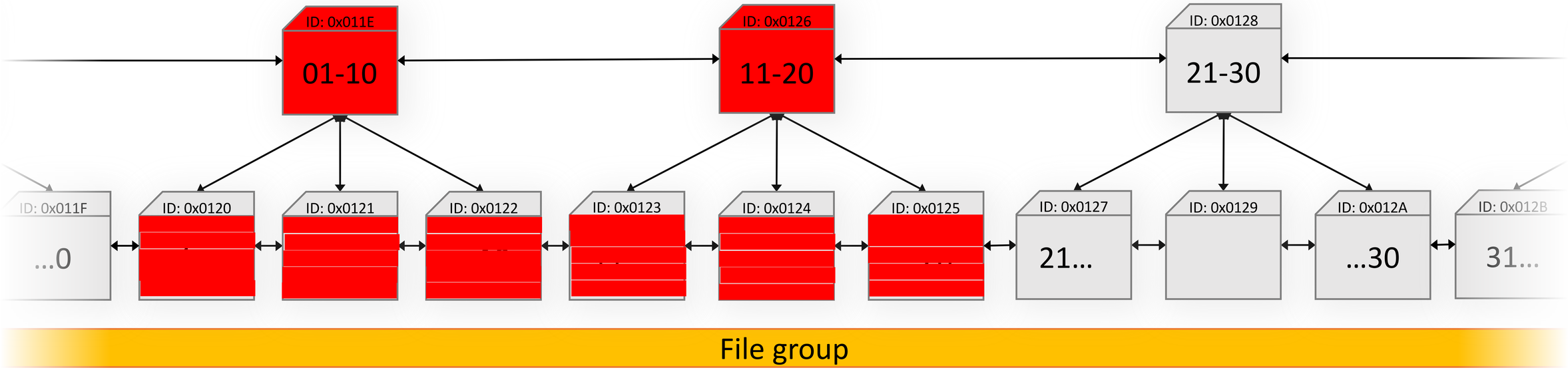
PASS Data Community SUMMIT 2022

A regular delete



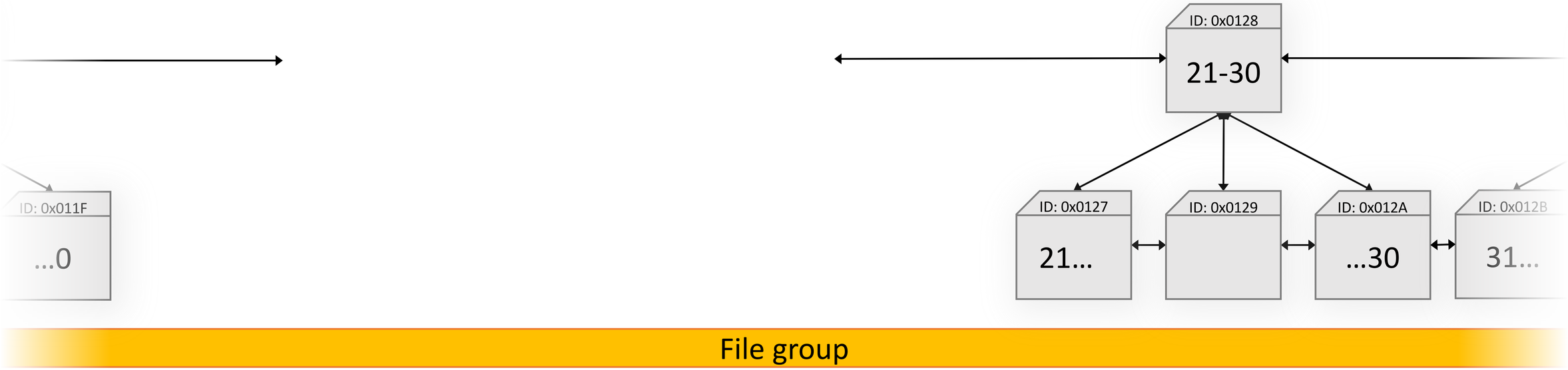
PASS Data Community SUMMIT 2022

A regular delete



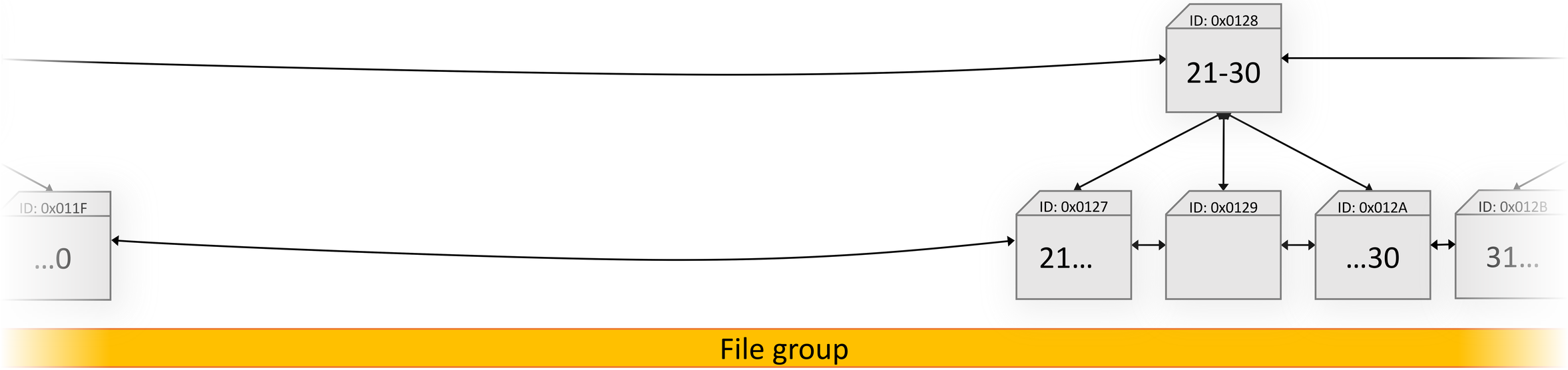
PASS Data Community SUMMIT 2022

A regular delete



PASS Data Community SUMMIT 2022

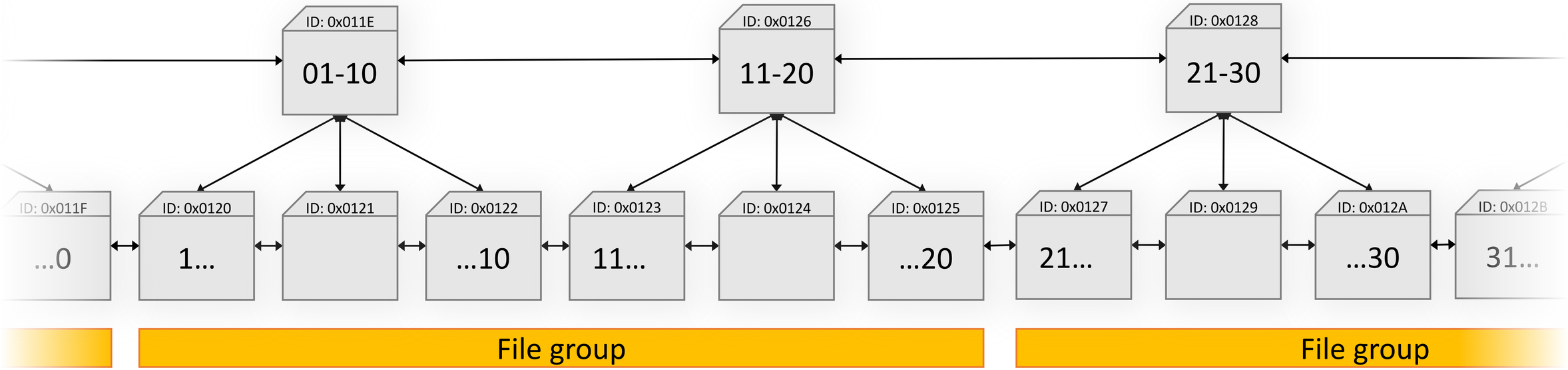
A regular delete



PASS Data Community SUMMIT 2022

Truncating a partition

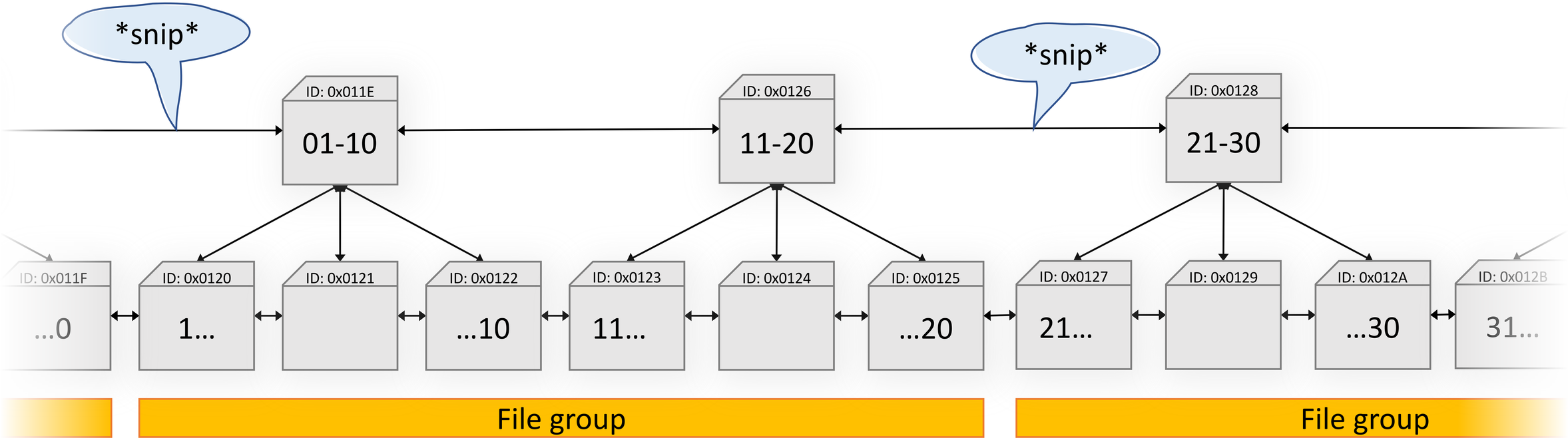
```
TRUNCATE TABLE dbo.partitioned_table
WITH (PARTITIONS (2));
```



* Truncating individual partitions is available on SQL Server 2016+

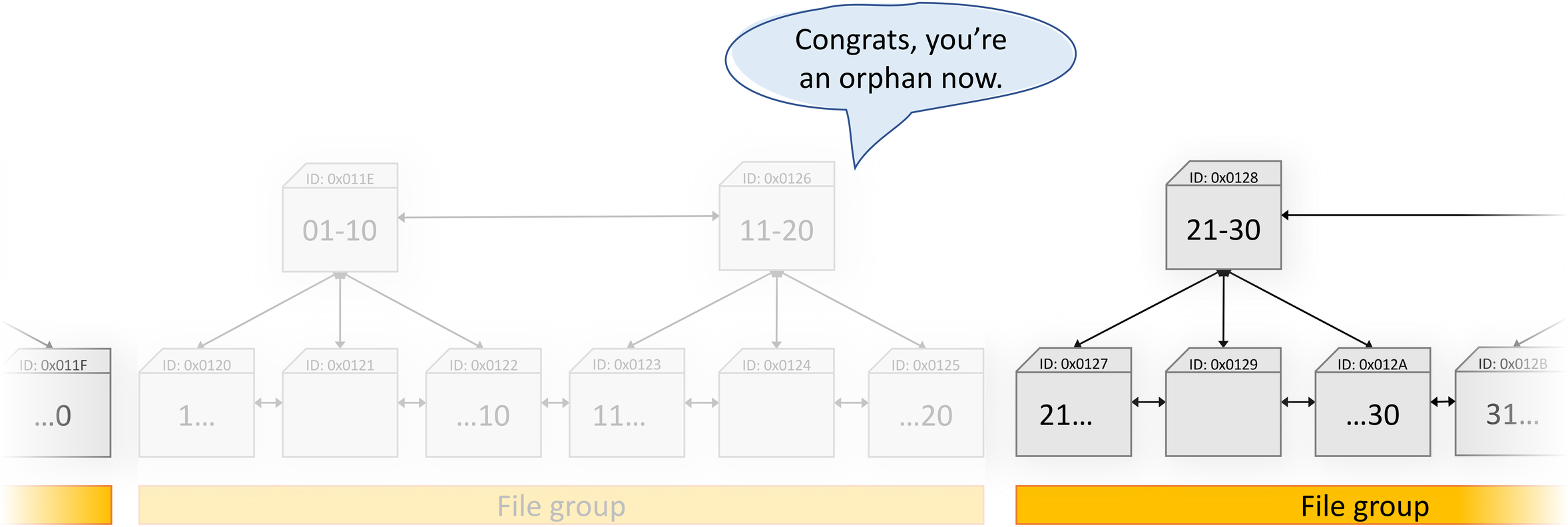
PASS Data Community SUMMIT 2022

Truncating a partition



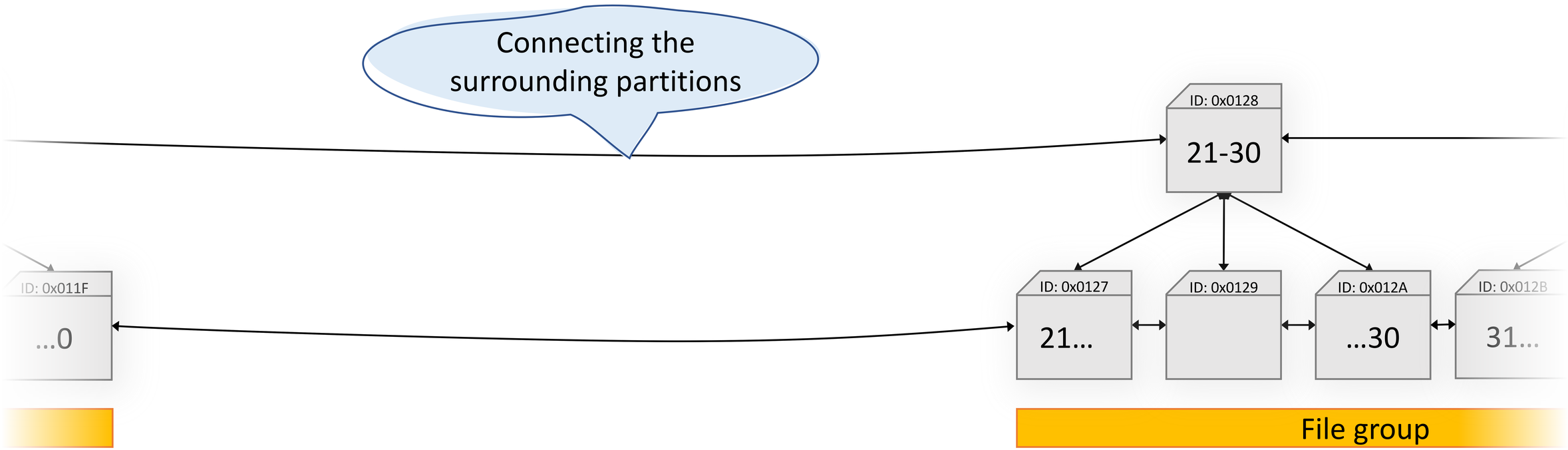
PASS Data Community SUMMIT 2022

Truncating a partition



PASS Data Community SUMMIT 2022

Truncating a partition



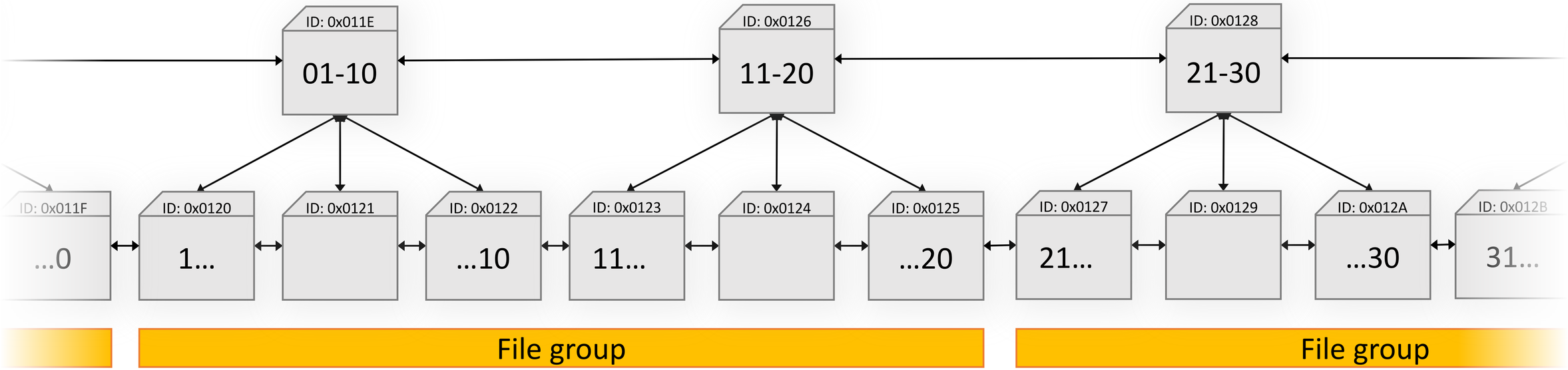
PASS Data Community SUMMIT 2022

DEMO

Image credit: Robert Ritchie, Unsplash

Switching a partition

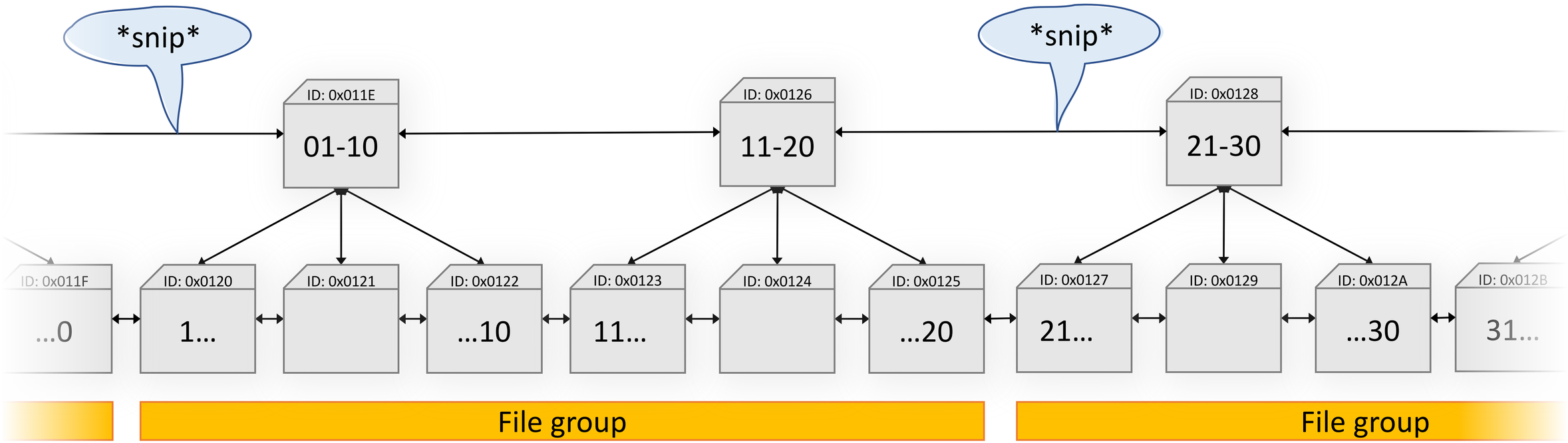
```
ALTER TABLE dbo.partitioned_table
SWITCH PARTITION 2 TO dbo.regular_table;
```



PASS Data Community SUMMIT 2022

Switching a partition

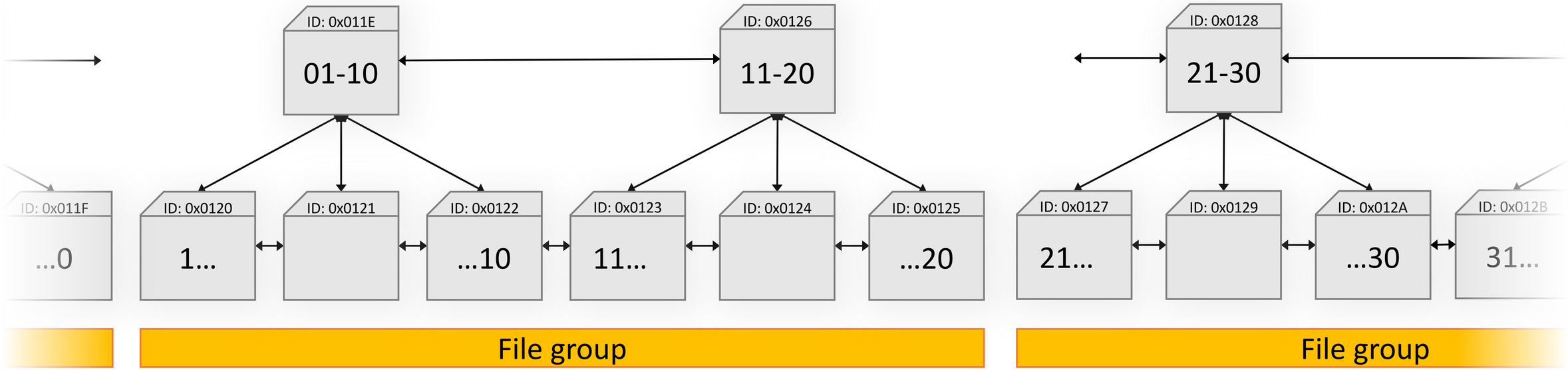
```
ALTER TABLE dbo.partitioned_table
SWITCH PARTITION 2 TO dbo.regular_table;
```



PASS Data Community SUMMIT 2022

Switching a partition

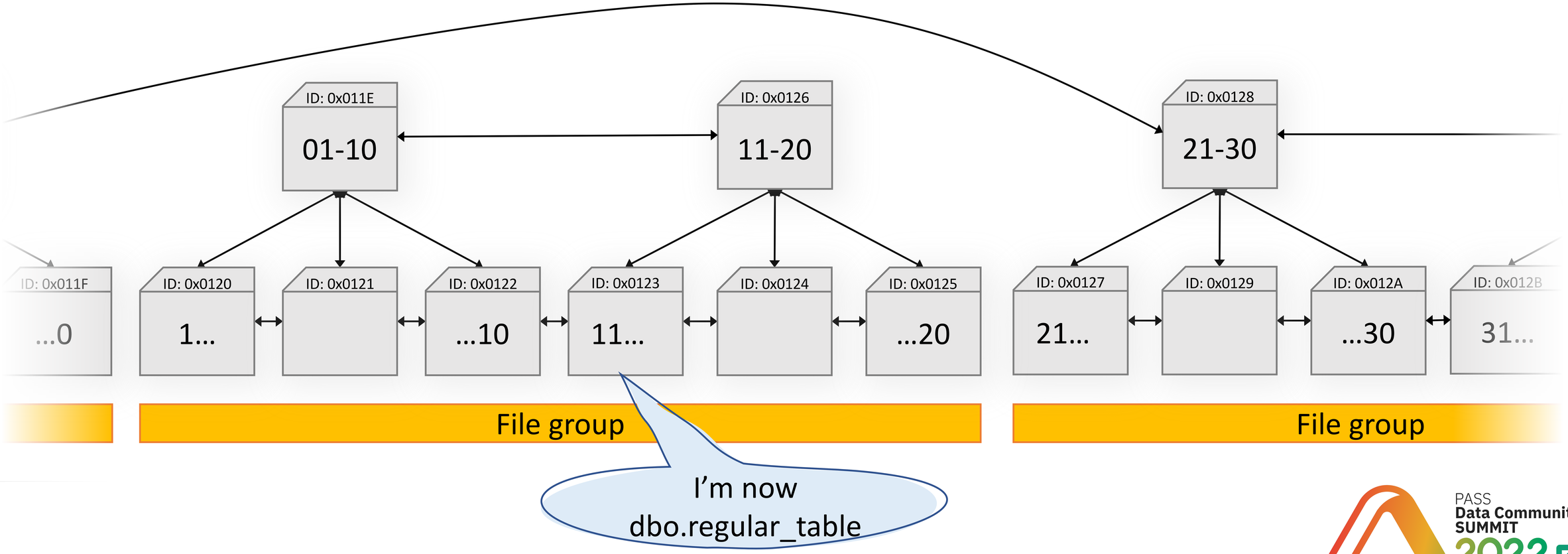
```
ALTER TABLE dbo.partitioned_table
SWITCH PARTITION 2 TO dbo.regular_table;
```



PASS Data Community SUMMIT 2022

Switching a partition

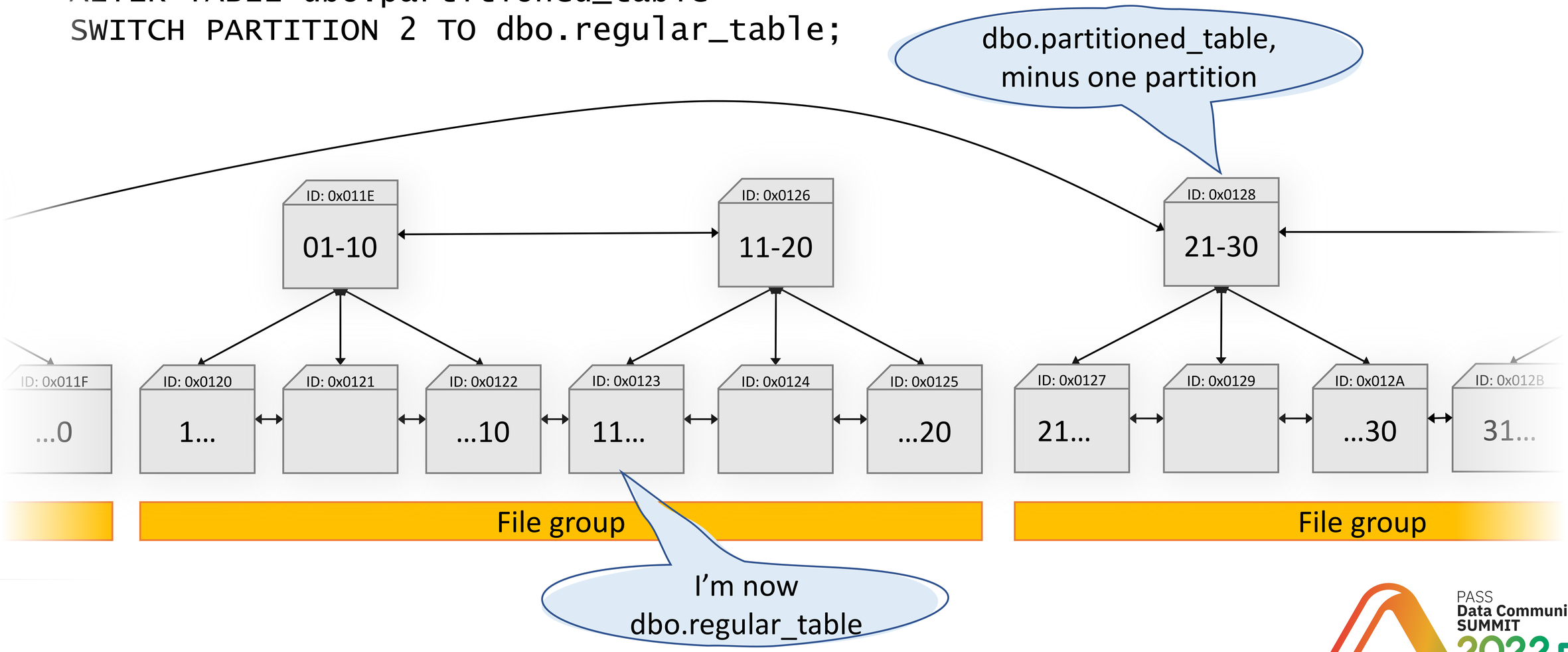
```
ALTER TABLE dbo.partitioned_table
SWITCH PARTITION 2 TO dbo.regular_table;
```



PASS Data Community SUMMIT 2022

Switching a partition

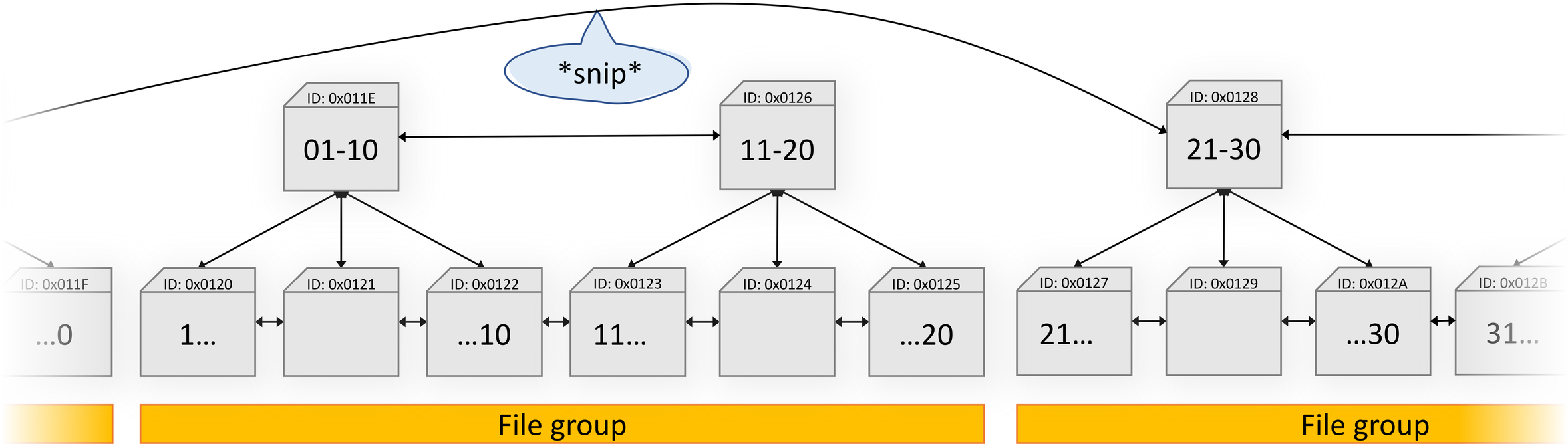
```
ALTER TABLE dbo.partitioned_table
SWITCH PARTITION 2 TO dbo.regular_table;
```



PASS Data Community SUMMIT 2022

.. and back

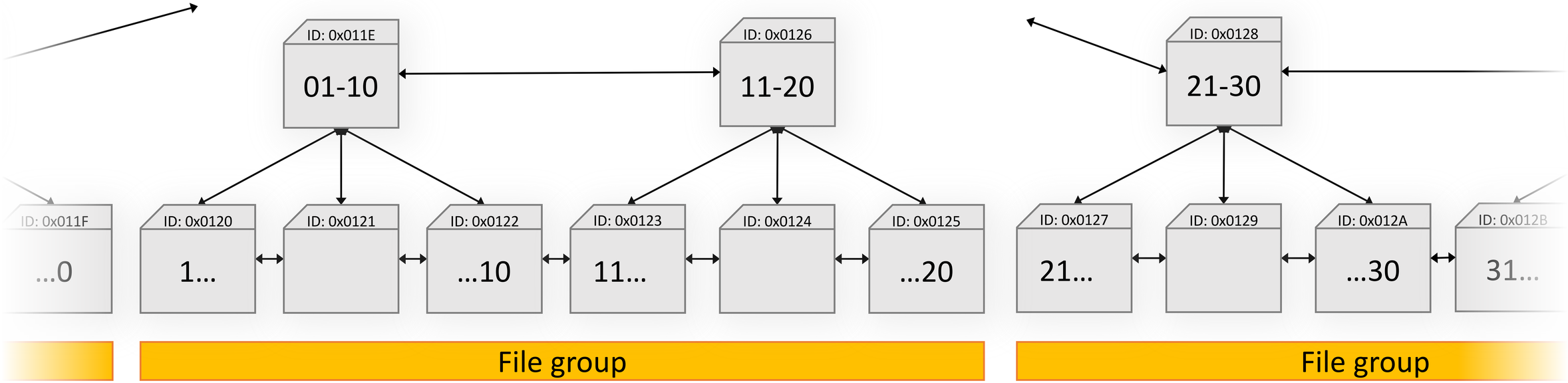
```
ALTER TABLE dbo.regular_table
SWITCH TO dbo.partitioned_table PARTITION 2;
```



PASS Data Community SUMMIT 2022

.. and back

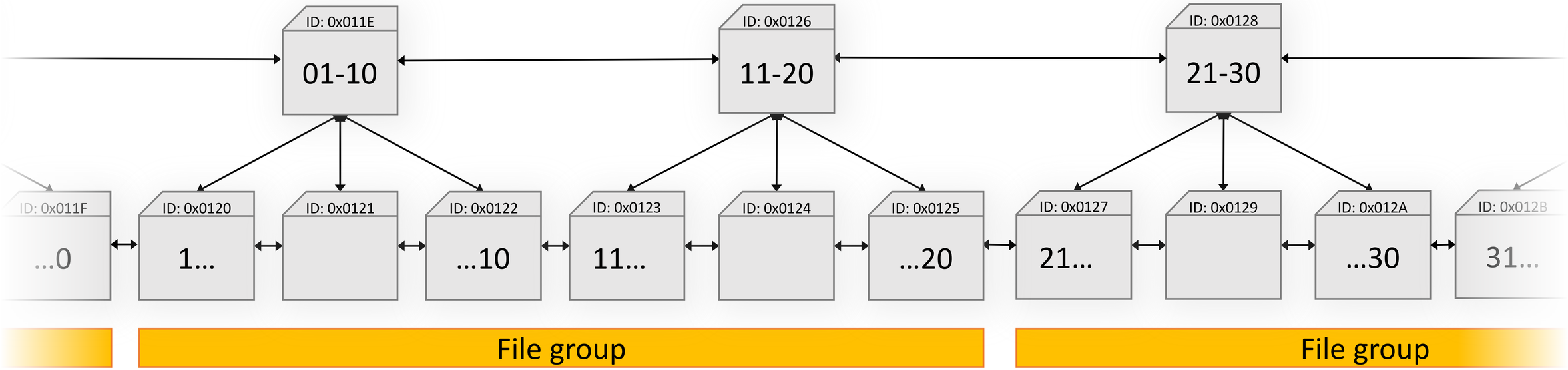
```
ALTER TABLE dbo.regular_table
SWITCH TO dbo.partitioned_table PARTITION 2;
```



PASS Data Community SUMMIT 2022

.. and back

```
ALTER TABLE dbo.regular_table
SWITCH TO dbo.partitioned_table PARTITION 2;
```



* Bonus fact: you can switch two *non-partitioned* tables, too!

PASS Data Community SUMMIT 2022

These are metadata operations!

- The data doesn’t move.
- Only the pointers are updated.
- Metadata operations require metadata permissions!

DEMO

Image credit: Zhifei Zhou, Unsplash

"Right is right"

… AS RANGE LEFT
FOR VALUES (100, 200, 300)

		$n \leq$	100
100	$<$	$n \leq$	200
200	$<$	$n \leq$	300
300	$<$	n	

… AS RANGE RIGHT
FOR VALUES (100, 200, 300)

		$n <$	100
100	$\leq$	$n <$	200
200	$\leq$	$n <$	300
300	$\leq$	n	

PASS Data Community SUMMIT 2022

"Right is right"

```
CREATE PARTITION FUNCTION AnnualFunction(date)
AS RANGE RIGHT
        FOR VALUES ( '2020-01-01' , '2021-01-01' , '2022-01-01' );

CREATE PARTITION SCHEME Annual
AS PARTITION AnnualFunction
TO ( [FILEGROUP_A] , [FILEGROUP_B] , [FILEGROUP_C] , [FILEGROUP_D] );
```

< 2020-01-01

≥ 2020-01-01
< 2021-01-01

PASS Data Community SUMMIT 2022

"Right is right"

```
CREATE PARTITION FUNCTION AnnualFunction(date)
AS RANGE LEFT
        FOR VALUES ( '2020-01-01' , '2021-01-01' , '2022-01-01' );

CREATE PARTITION SCHEME Annual
AS PARTITION AnnualFunction
                TO ( [FILEGROUP_A] , [FILEGROUP_B] , [FILEGROUP_C] , [..._D] );
```

≤ 2020-01-01

> 2020-01-01
≤ 2021-01-01

PASS Data Community SUMMIT 2022

Split/merge

- You can *split* an existing partition into two, by adding a new boundary.

… 2018	2019	2020	2021…
Filegroup A	Filegroup B	Filegroup C	Filegroup D

PASS Data Community SUMMIT 2022

Split/merge

- You can *split* an existing partition into two, by adding a new boundary.

PASS Data Community SUMMIT 2022

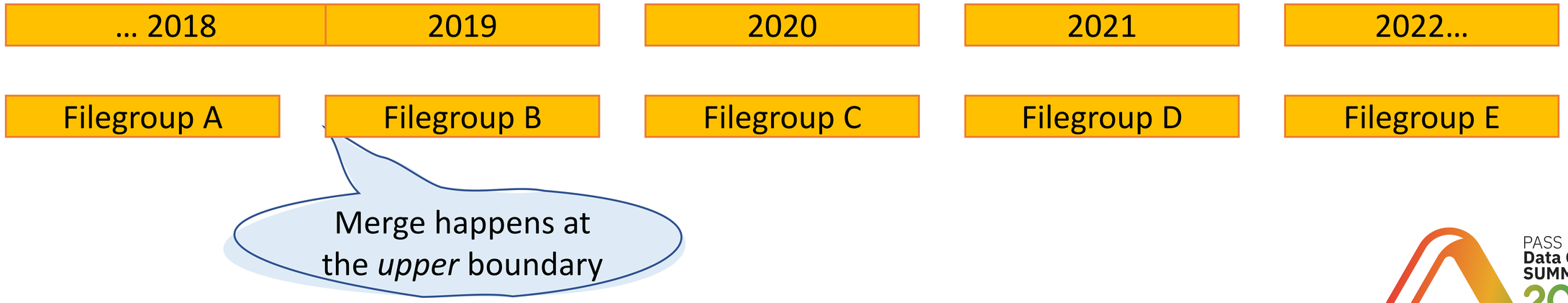
Split/merge

- You can *split* an existing partition into two, by adding a new boundary.

PASS Data Community SUMMIT 2022

Split/merge

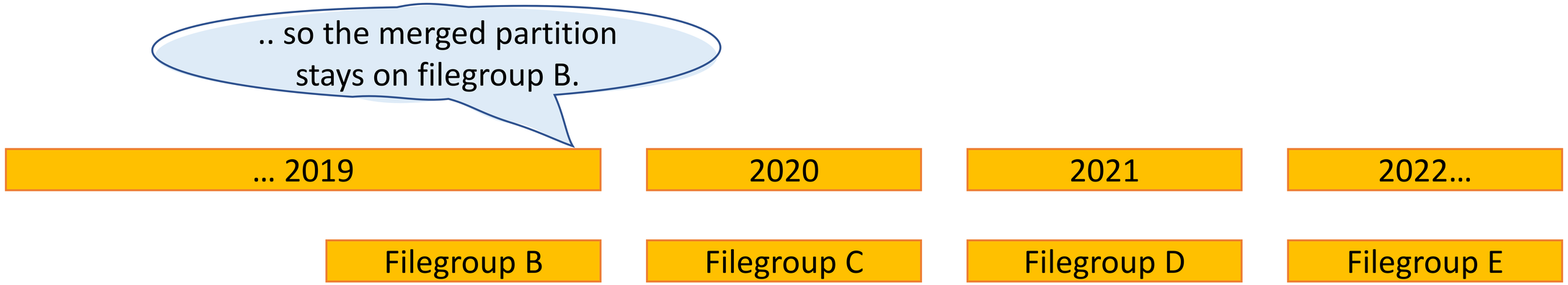
- You can *split* an existing partition into two, by adding a new boundary.
- Two partitions can also be *merged* into one, by removing a boundary.



PASS Data Community SUMMIT 2022

Split/merge

- You can *split* an existing partition into two, by adding a new boundary.
- Two partitions can also be *merged* into one, by removing a boundary.



PASS Data Community SUMMIT 2022

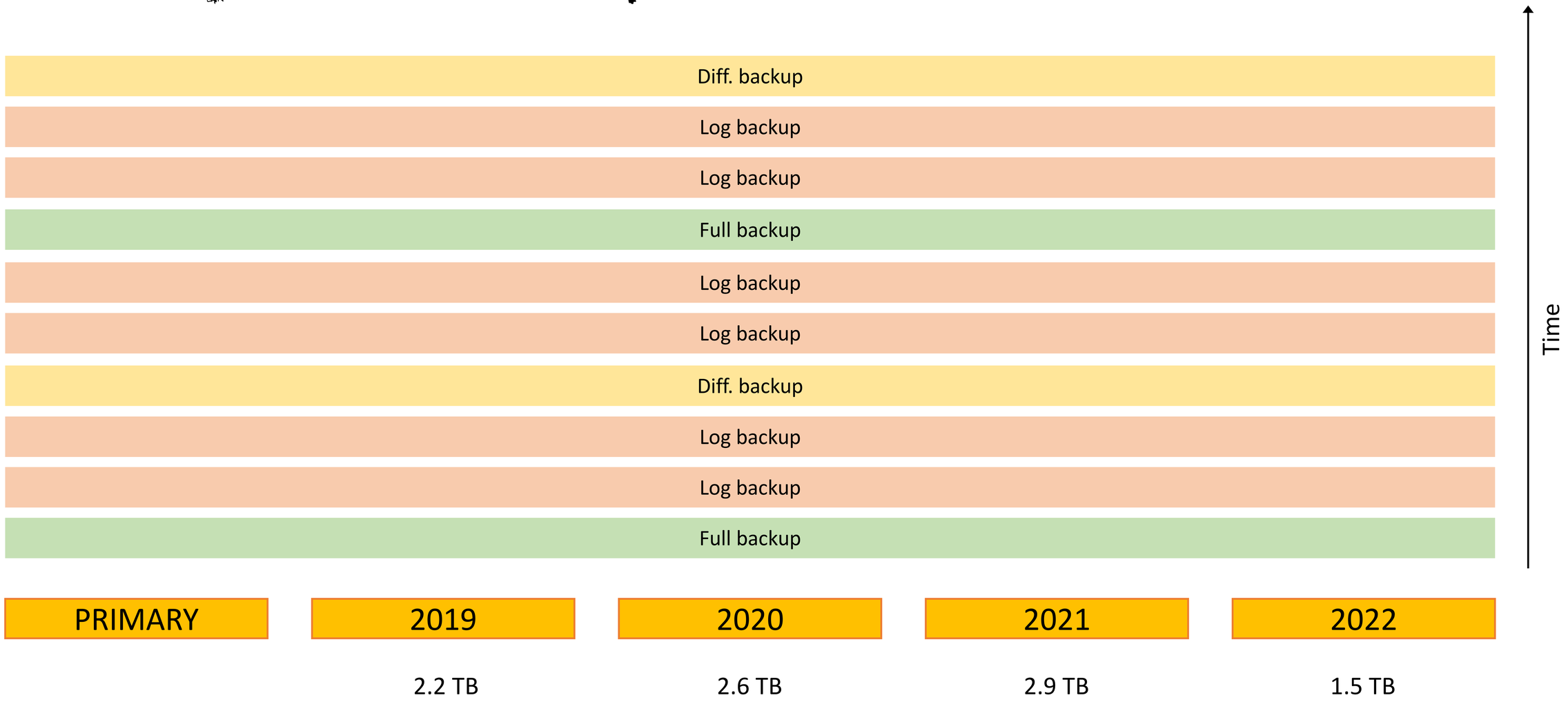
DEMO

Image credit: Zhifei Zhou, Unsplash

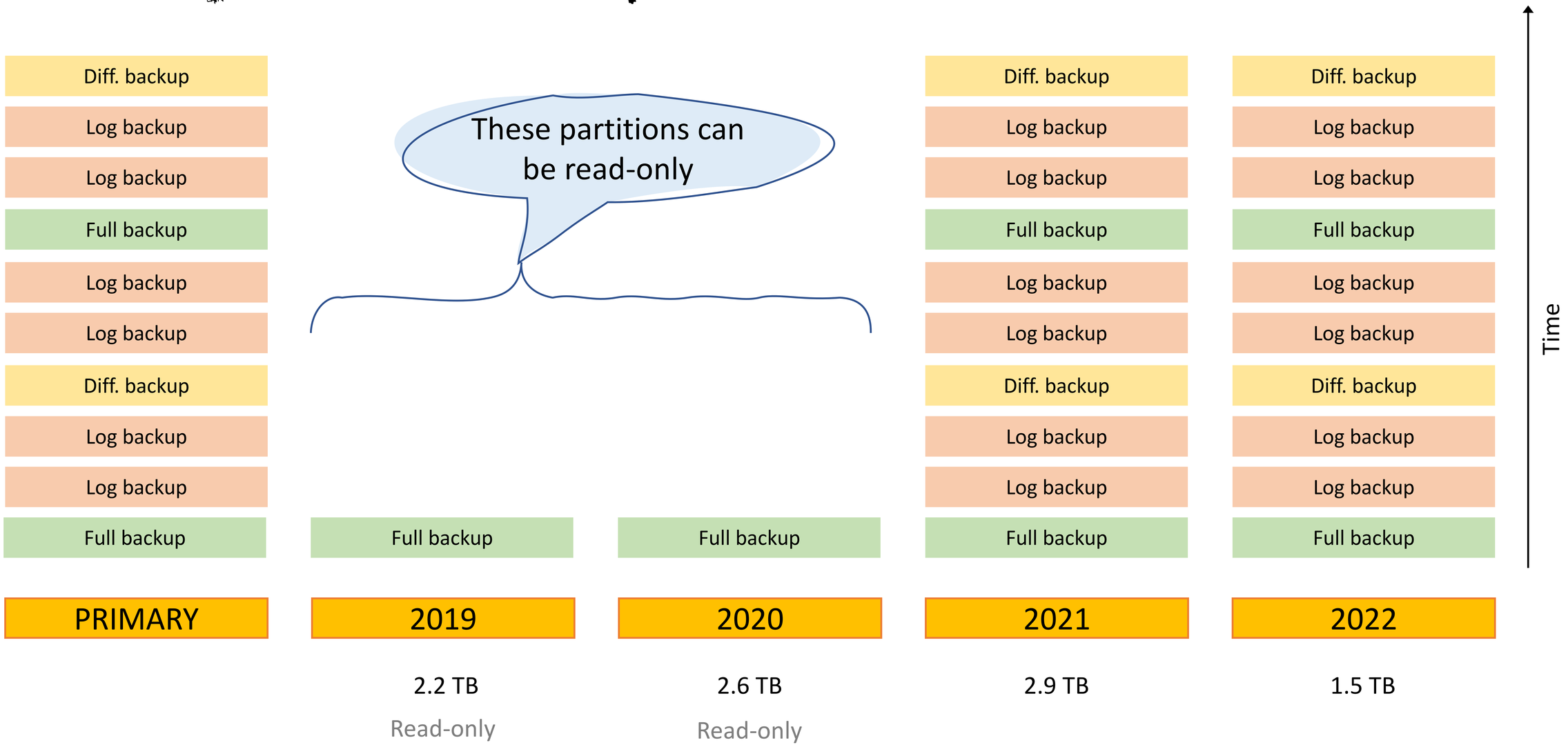
Aligned / non-aligned partitions

- If the table and its indexes all share the same partition scheme, the indexes are “aligned”
- If one or more indexes are not partitioned, or differently partitioned, they are “non-aligned”, and you won’t be able to perform partition-level operations like SPLIT, MERGE, TRUNCATE, SWITCH.

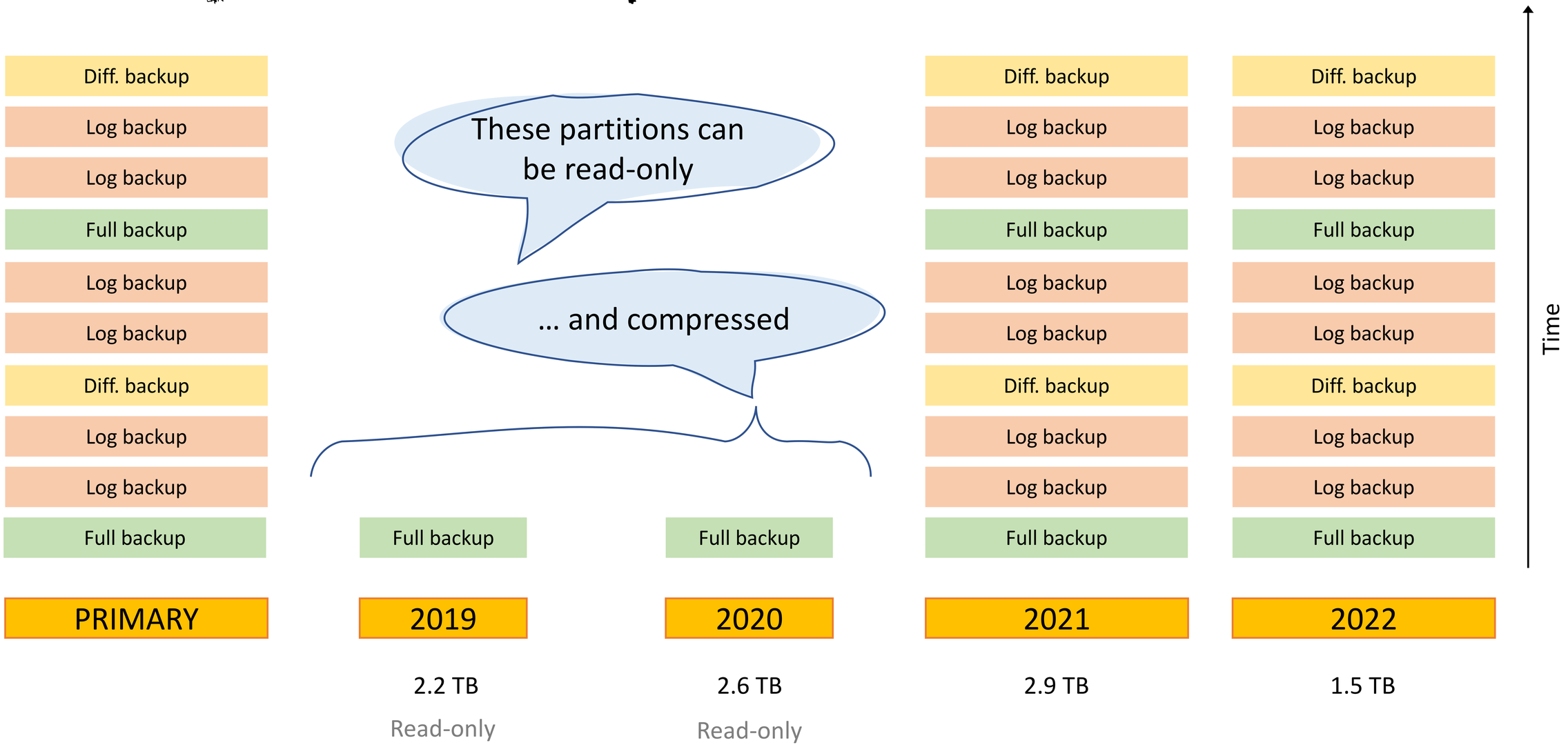
Make your backups smaller



Make your backups smaller



Make your backups smaller



Piecemeal restore (full)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups

PRIMARY	Some file	2019	2020	2021	2022	2023
		Read-only	Read-only			

Piecemeal restore (full)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups

Full backup, partial, recovery

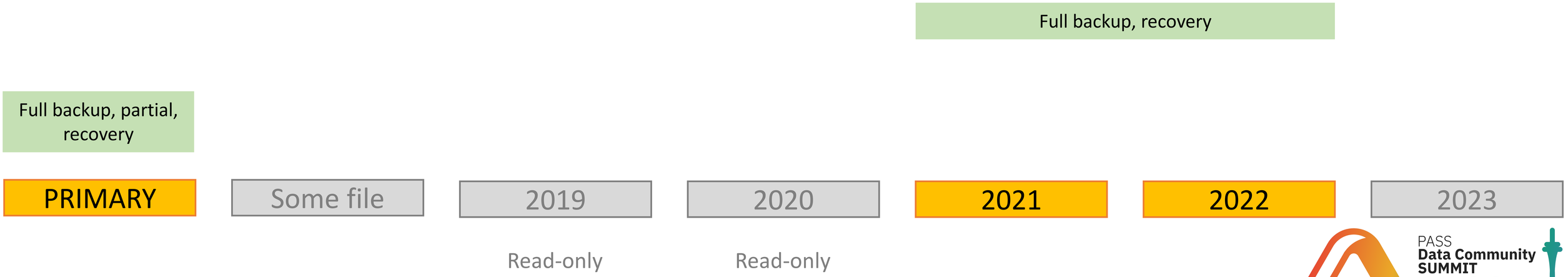
PRIMARY	Some file	2019	2020	2021	2022	2023
		Read-only	Read-only			

PASS Data Community SUMMIT 2022

Piecemeal restore (full)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups

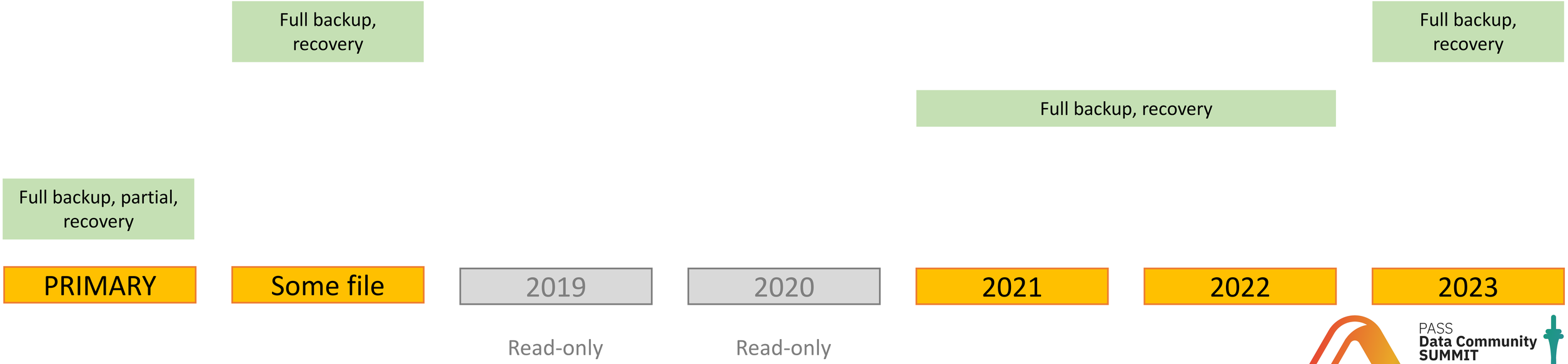


PASS Data Community SUMMIT 2022

Piecemeal restore (full)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups



PASS Data Community SUMMIT 2022

Piecemeal restore (full)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups

This can be a really old backup

Full backup, recovery

Full backup, recovery

Full backup, recovery

Full backup, recovery

Full backup, partial, recovery

PRIMARY	Some file	2019	2020	2021	2022	2023
		Read-only	Read-only			

PASS Data Community SUMMIT 2022

DEMO

Image credit: Timothy Eberly, Unsplash

Piecemeal restore (full + log)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups
- Transaction log backup(s) for the read-write filegroups

PRIMARY	Some file	2019	2020	2021	2022	2023
		Read-only	Read-only			

PASS Data Community SUMMIT 2022

Piecemeal restore (full + log)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups
- Transaction log backup(s) for the read-write filegroups

Full, partial, norec.

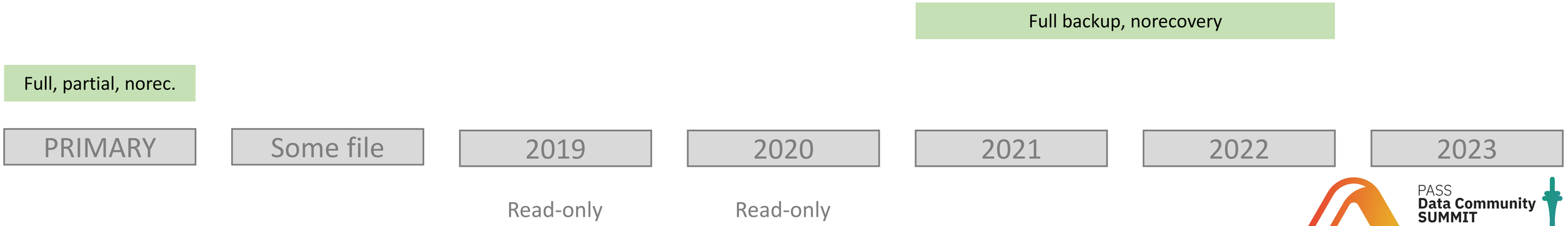
PRIMARY	Some file	2019	2020	2021	2022	2023
		Read-only	Read-only			

PASS Data Community SUMMIT 2022

Piecemeal restore (full + log)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups
- Transaction log backup(s) for the read-write filegroups

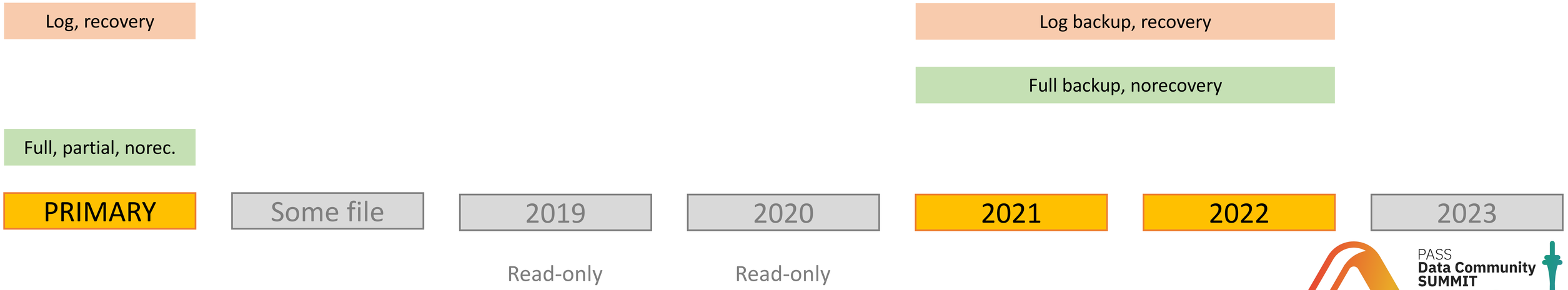


PASS Data Community SUMMIT 2022

Piecemeal restore (full + log)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups
- Transaction log backup(s) for the read-write filegroups

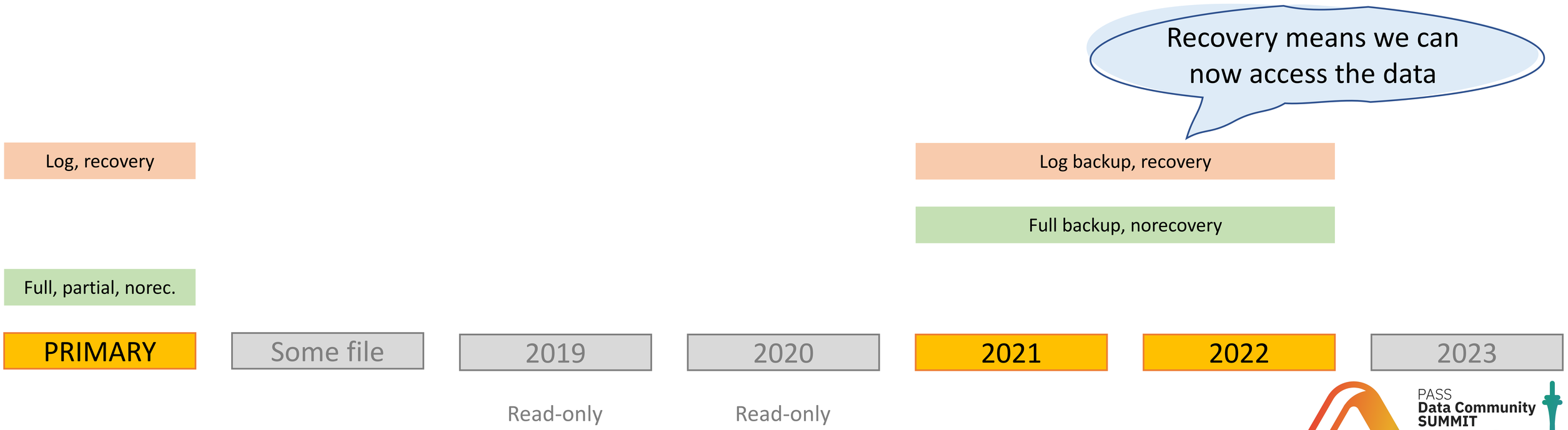


PASS Data Community SUMMIT 2022

Piecemeal restore (full + log)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups
- Transaction log backup(s) for the read-write filegroups

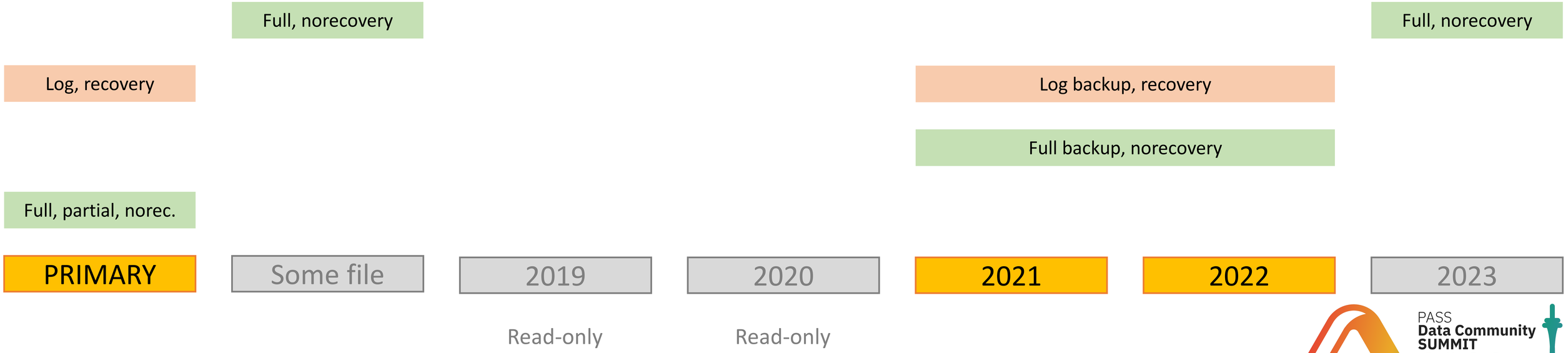


PASS Data Community SUMMIT 2022

Piecemeal restore (full + log)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups
- Transaction log backup(s) for the read-write filegroups

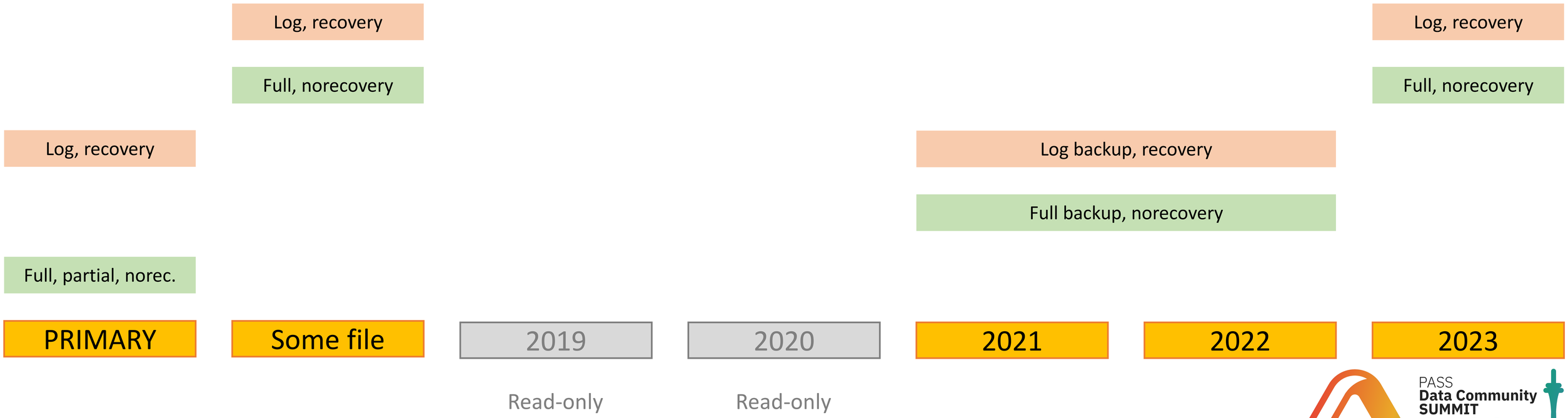


PASS Data Community SUMMIT 2022

Piecemeal restore (full + log)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups
- Transaction log backup(s) for the read-write filegroups

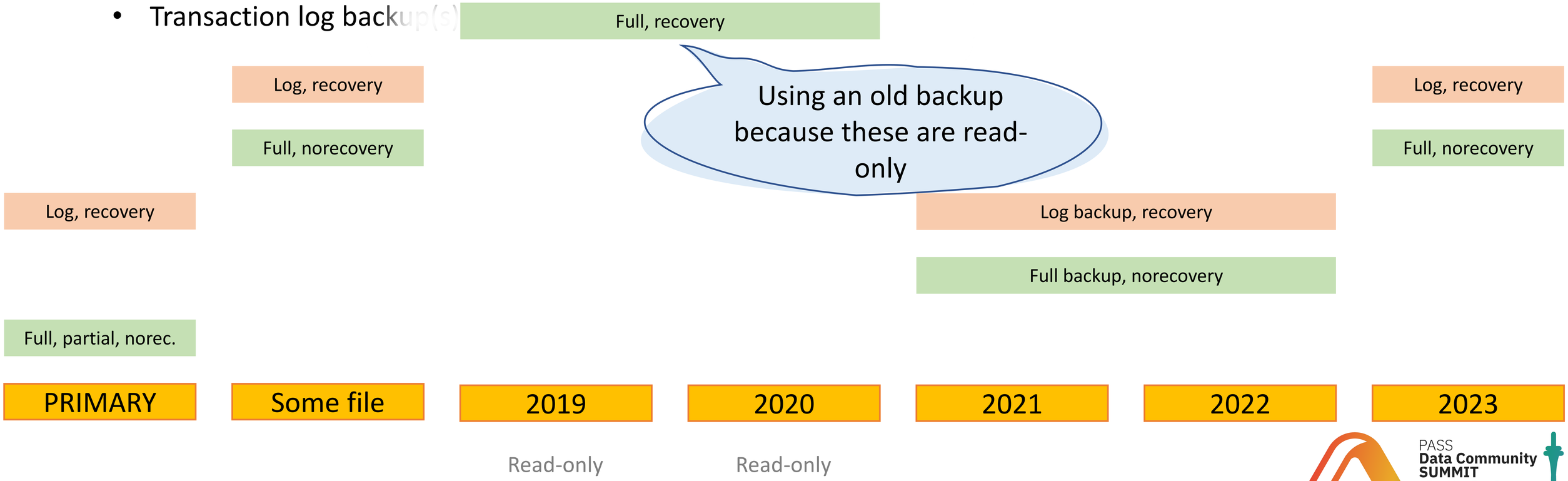


PASS Data Community SUMMIT 2022

Piecemeal restore (full + log)

We have:

- An old FULL backup of the read-only filegroups
- A recent FULL backup of the read-write filegroups
- Transaction log backup(s)



PASS Data Community SUMMIT 2022

DEMO

Image credit: Eric Zhang, Unsplash

Seeks & scans work differently

```
SELECT AccountID, SUM(Amount) AS Net_balance_change
FROM dbo.AccountTransactions
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
GROUP BY AccountID;
```

Seeks & scans work differently

```
SELECT AccountID, SUM(Amount) AS Net_balance_change
FROM dbo.AccountTransactions
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
GROUP BY AccountID;
```

AccountID
TransactionDate
TransactionID

Seeks & scans work differently

```
SELECT AccountID, SUM(Amount) AS Net_balance_change
FROM dbo.AccountTransactions
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
GROUP BY AccountID;
```

AccountID
TransactionDate
TransactionID

YEAR(TransactionDate)
AccountID
TransactionDate
TransactionID

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

AccountID
TransactionDate
TransactionID

YEAR(TransactionDate)
AccountID
TransactionDate
TransactionID

29 600 pages

- Seek AccountID 810000000-820000000

PASS Data Community SUMMIT 2022

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

AccountID
TransactionDate
TransactionID

YEAR(TransactionDate)
AccountID
TransactionDate
TransactionID

29 600 pages

- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2022-06

PASS Data Community SUMMIT 2022

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

AccountID
TransactionDate
TransactionID

YEAR(TransactionDate)
AccountID
TransactionDate
TransactionID

29 600 pages

- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2022-06
- Aggregate on AccountID

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

AccountID
TransactionDate
TransactionID

29 600 pages

- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2022-06
- Aggregate on AccountID

YEAR(TransactionDate)
AccountID
TransactionDate
TransactionID

1 800 pages

- For partition 2021
- Seek AccountID 810000000-820000000

PASS Data Community SUMMIT 2022

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

AccountID
TransactionDate
TransactionID

29 600 pages

- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2022-06
- Aggregate on AccountID

YEAR(TransactionDate)
AccountID
TransactionDate
TransactionID

1 800 pages

- For partition 2021
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2021-12

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

AccountID
TransactionDate
TransactionID

29 600 pages

- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2022-06
- Aggregate on AccountID

YEAR(TransactionDate)
AccountID
TransactionDate
TransactionID

1 800 pages

- For partition 2021
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2021-12
- Aggregate on AccountID

PASS Data Community SUMMIT 2022

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

AccountID
TransactionDate
TransactionID

29 600 pages

- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2022-06
- Aggregate on AccountID

YEAR(TransactionDate)
AccountID
TransactionDate
TransactionID

1 800 pages

- For partition 2021
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2021-12
- Aggregate on AccountID

600 pages

- For partition 2022
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2022-01 – 2022-06
- Aggregate on AccountID

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

- Seek AccountID 810000000-820000000

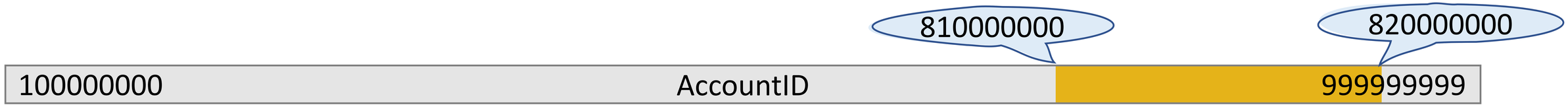
100000000	AccountID	999999999

PASS Data Community SUMMIT 2022

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

- Seek AccountID 810000000-820000000

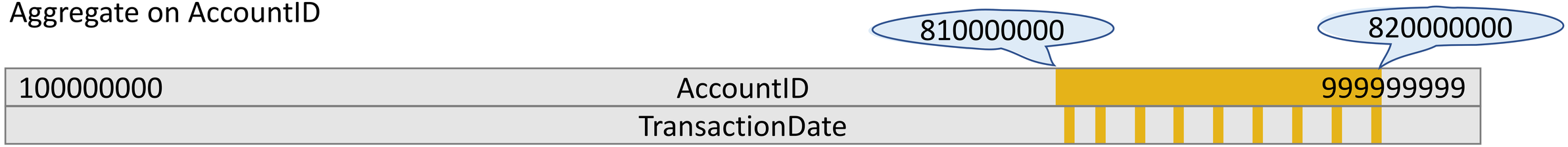


PASS Data Community SUMMIT 2022

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

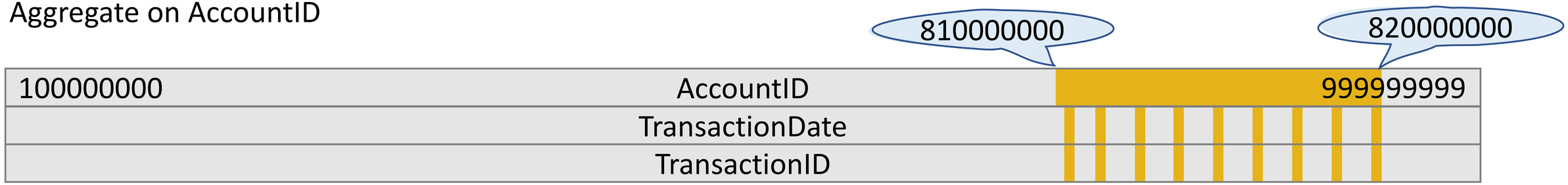
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2022-06
- Aggregate on AccountID



Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

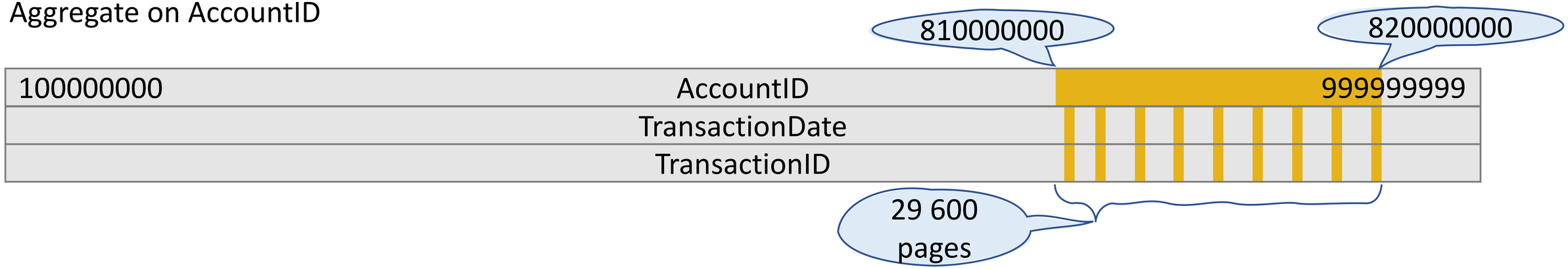
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2022-06
- Aggregate on AccountID



Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

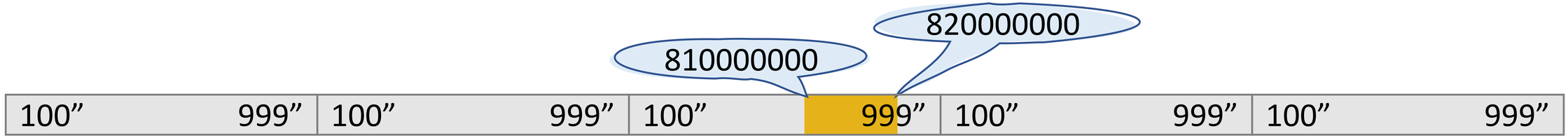
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2022-06
- Aggregate on AccountID



Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

- For partition 2021
- Seek AccountID 810000000-820000000

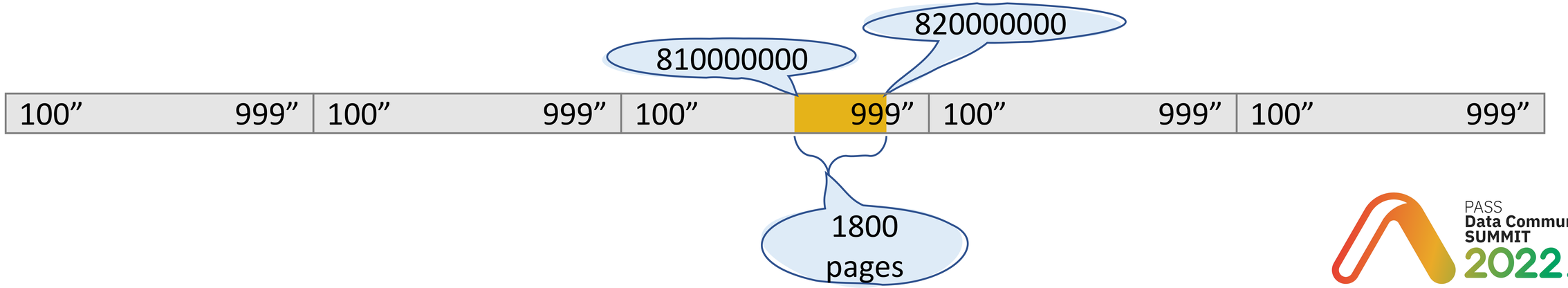


PASS Data Community SUMMIT 2022

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

- For partition 2021
- Seek AccountID 810000000-820000000

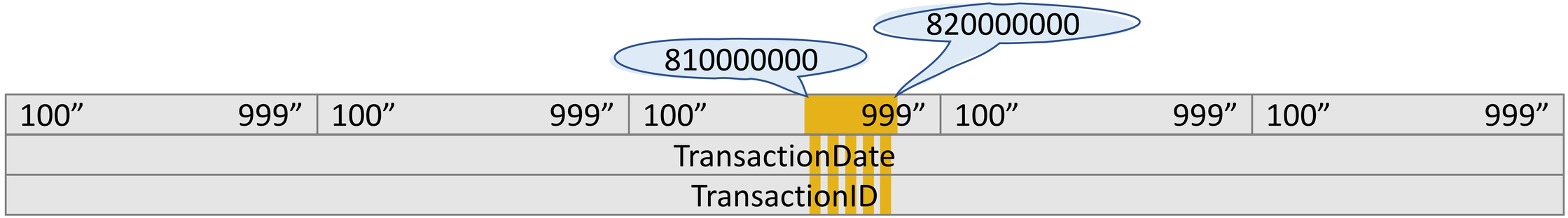


PASS
Data Community
SUMMIT
2022

Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

- For partition 2021
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2021-12
- Aggregate on AccountID

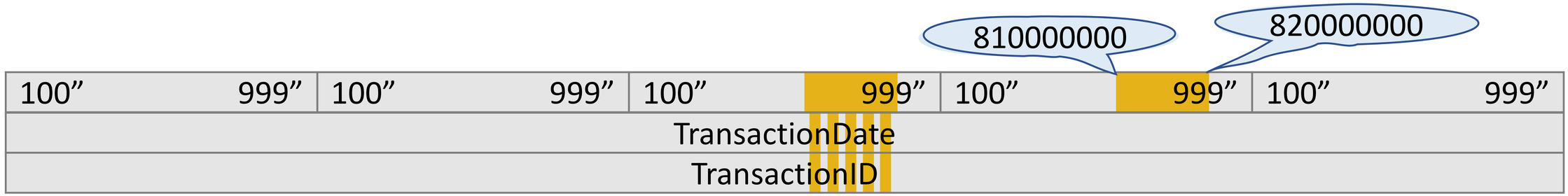


Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

- For partition 2021
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2021-12
- Aggregate on AccountID

- For partition 2022
- Seek AccountID 810000000-820000000

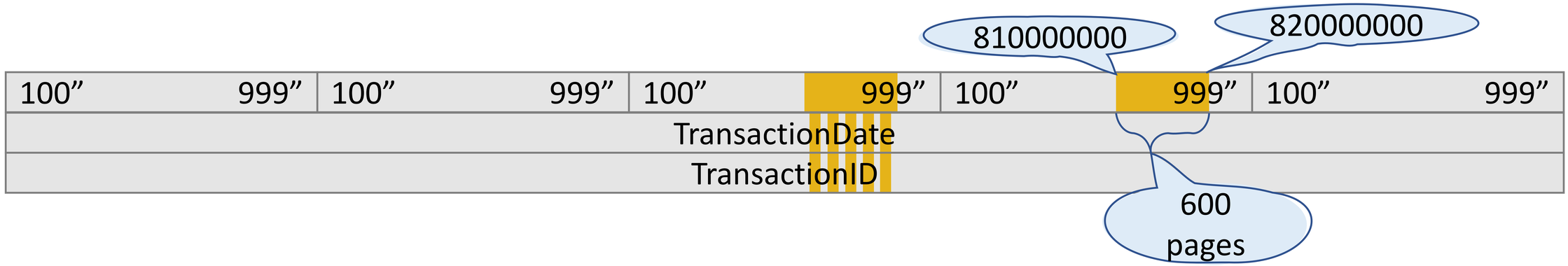


Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

- For partition 2021
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2021-12
- Aggregate on AccountID

- For partition 2022
- Seek AccountID 810000000-820000000

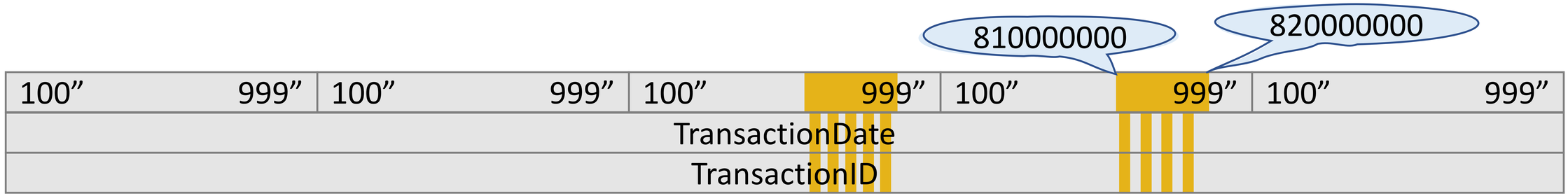


Seeks & scans work differently

```
WHERE TransactionDate BETWEEN '2021-07-01' AND '2022-06-30'
  AND AccountID BETWEEN 810000000 AND 820000000
```

- For partition 2021
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2021-12
- Aggregate on AccountID

- For partition 2022
- Seek AccountID 810000000-820000000
- Filter TransactionDate 2021-07 – 2021-12
- Aggregate on AccountID



DEMO

Image credit: Robert Ritchie, Unsplash

Takeaway: Why partitioning?

- Some costly DML operations can be turned into super-fast DDL statements
- Some queries can be tuned to use a single partition (partition elimination)
- Can solve specific latch contention issues by spreading writes
- You can backup only read-write partitions – faster backups
- You can restore the important partitions first – shorter RTO

… or maybe not?

- More complex to maintain
- More complex to develop
- Tuned for non-partitioned ≠ tuned for partitioning

Standing on the shoulders of giants

I found inspiration in blogs, presentations and more from:

- Kendra Little
- Cathrine Wilhelmsen
- Uwe Ricken

Session evaluation

Your feedback is important to us

Evaluate this session at:

www.PASSDataCommunitySummit.com/evaluation

PASS
Data Community
SUMMIT
2022

Thank you

Enjoy the rest of the conference!

Daniel Hutmacher

daniel@strd.co

@dhmacher

sqlsunday.com

PASS
Data Community
SUMMIT
2022