

SQLServerFast.com

Execution Plan Video Training

Block 2: Reading data

Level: Advanced

Chapter 2: Memory-optimized indexes

Memory-optimized indexes

Memory-optimized tables and indexes

Part of feature set “In-Memory OLTP” (aka “Hekaton”)

- Natively compiled stored procedures

- Lock- and latch-free storage structures

- Highly optimized transaction logging

- All data stored in memory

 - Indexes designed to benefit from being stored in memory

Memory-optimized indexes

Memory-optimized index structure

All data stored in memory

Row structure

								25	42	Belgium	Black	Sirius
--	--	--	--	--	--	--	--	----	----	---------	-------	--------

Actual data

Begin and end timestamp

Not actually a “time” stamp, but an internal transaction counter

Eight pointers to other rows

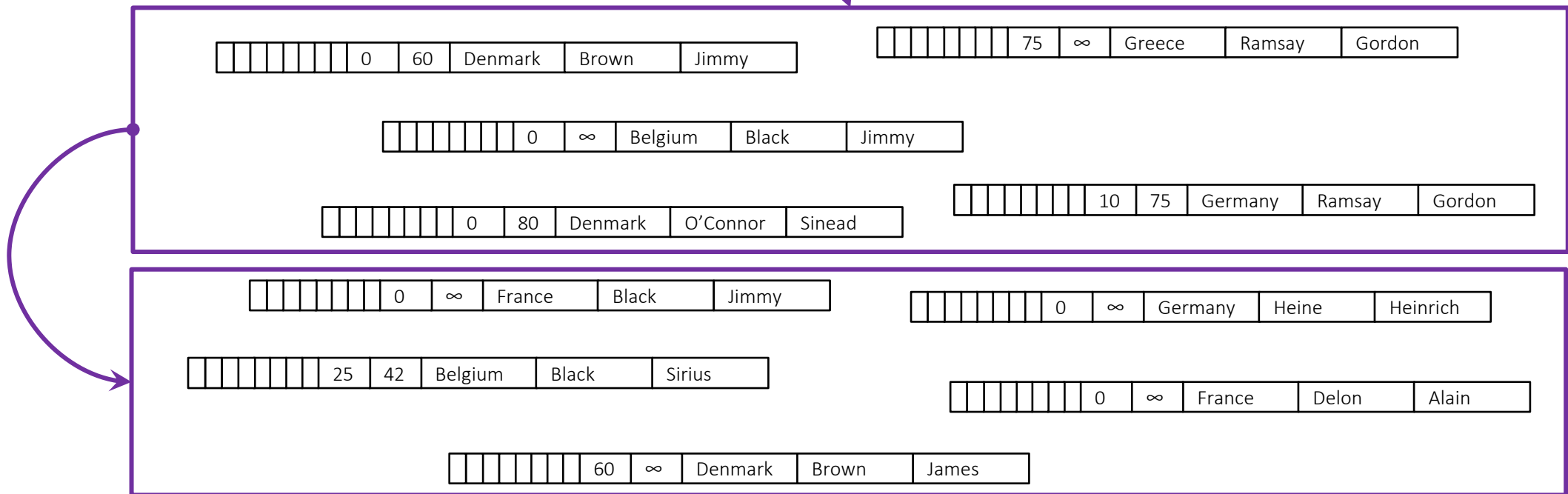
These implement the index structures

Memory-optimized indexes

Memory-optimized index structure

Varheap

First block

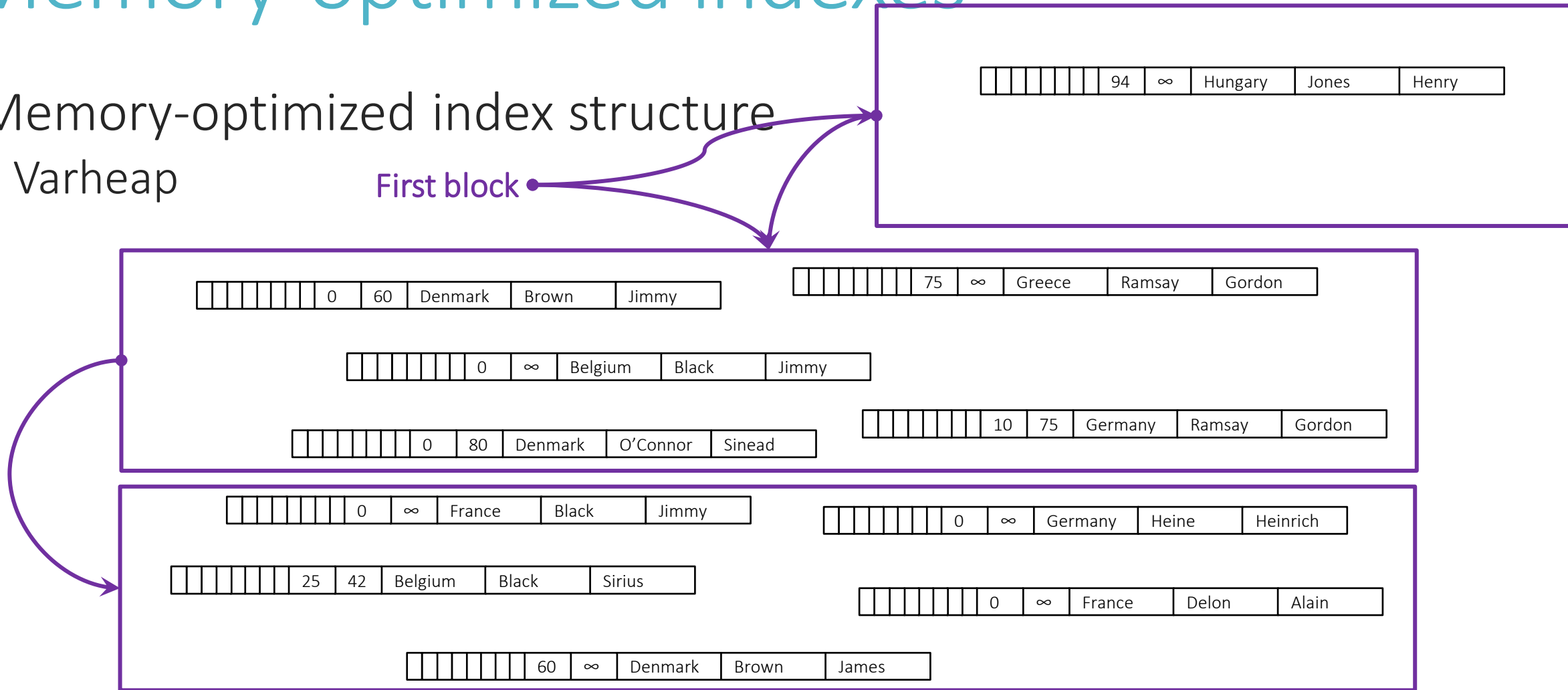


Memory-optimized indexes

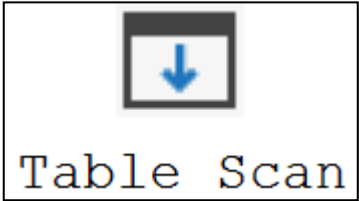
Memory-optimized index structure

Varheap

First block



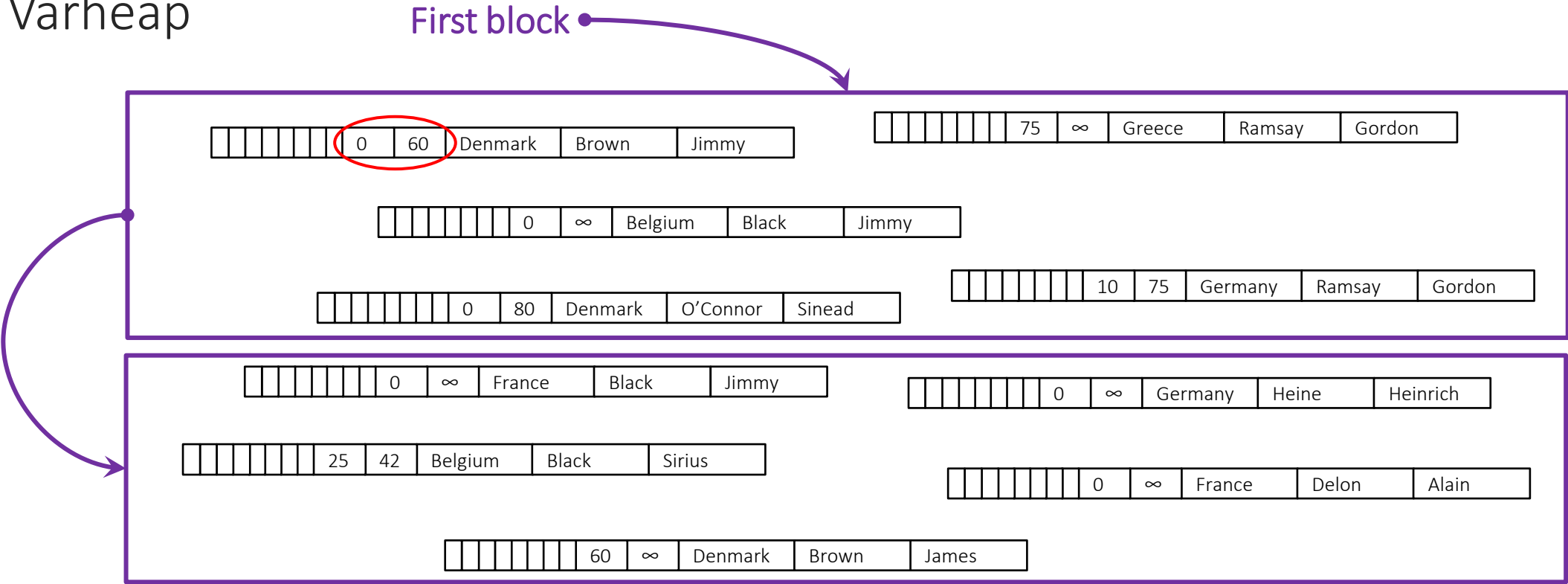
Memory-optimized indexes



Memory-optimized index structure

Varheap

First block

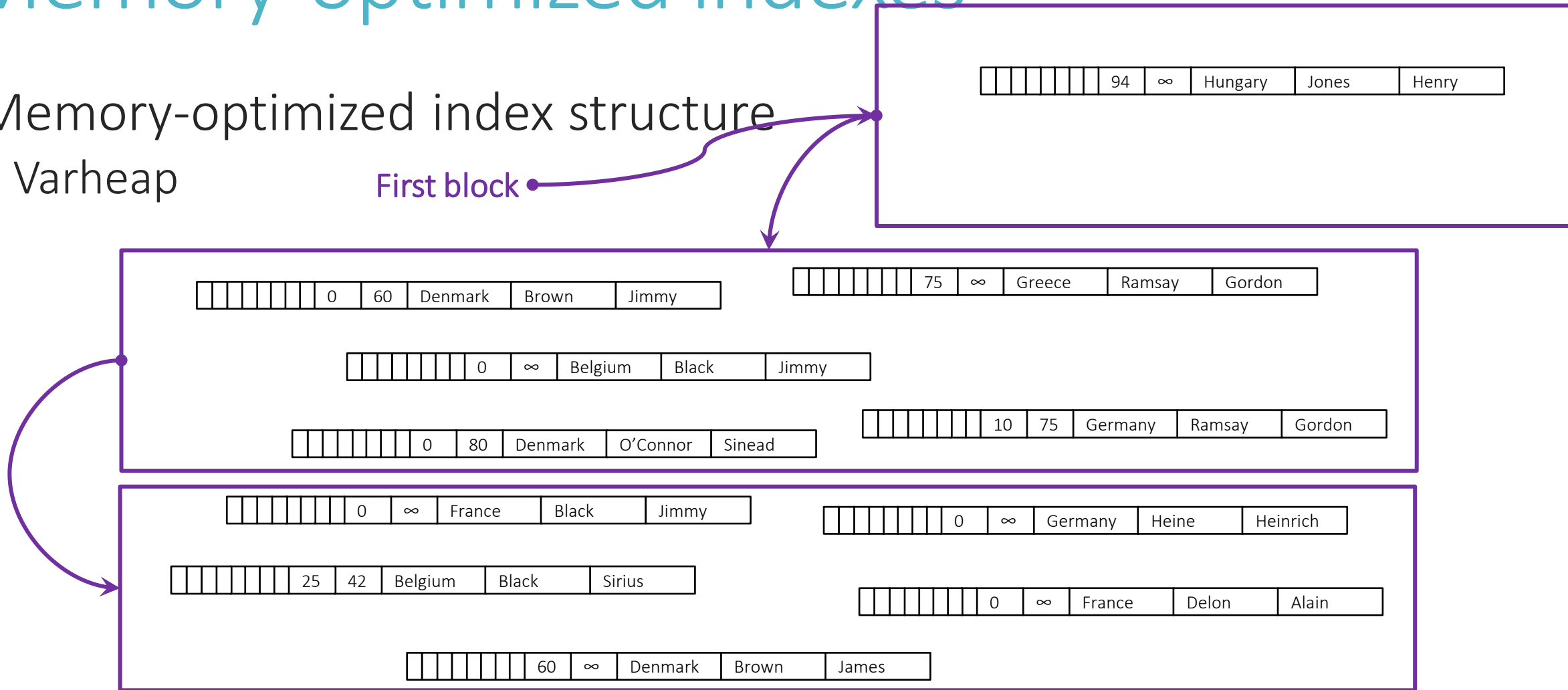


Memory-optimized indexes

Memory-optimized index structure

Varheap

First block



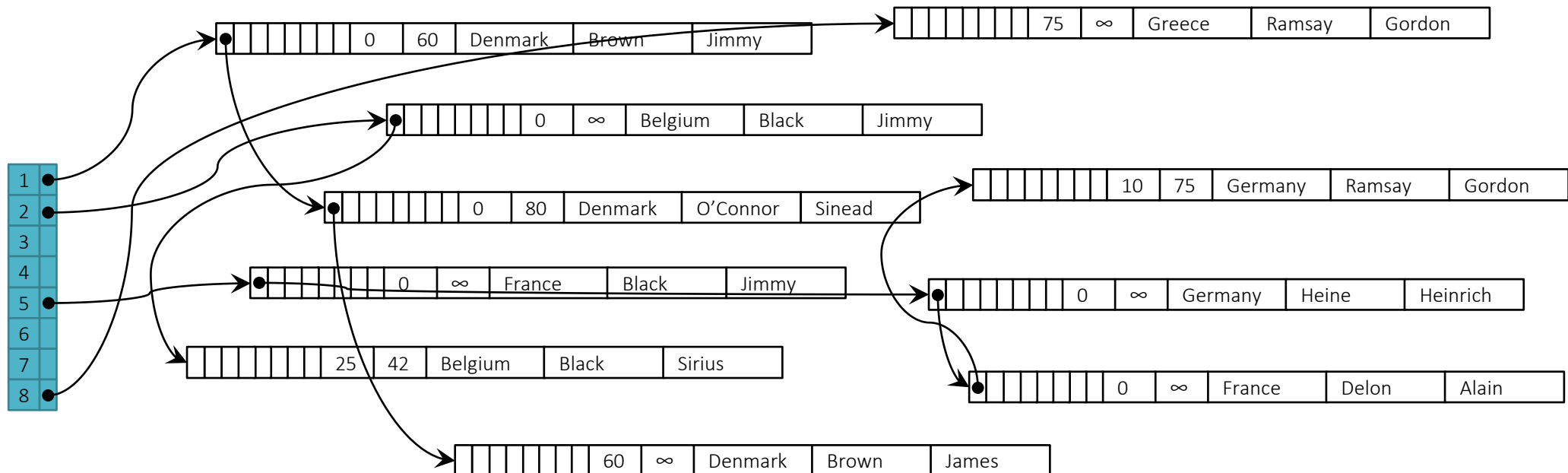
Memory-optimized indexes

```
CREATE TABLE dbo.SampleMemOptTable
(
  CountryName varchar(30) NOT NULL,
  LastName    varchar(60) NOT NULL,
  FirstName   varchar(50) NOT NULL,
  INDEX ix_Hash_CountryName
    NONCLUSTERED HASH (CountryName)
    WITH (BUCKET_COUNT = 8))
WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA);
```

Memory-optimized index structure

Memory-optimized nonclustered hash index

Short name: Hash index



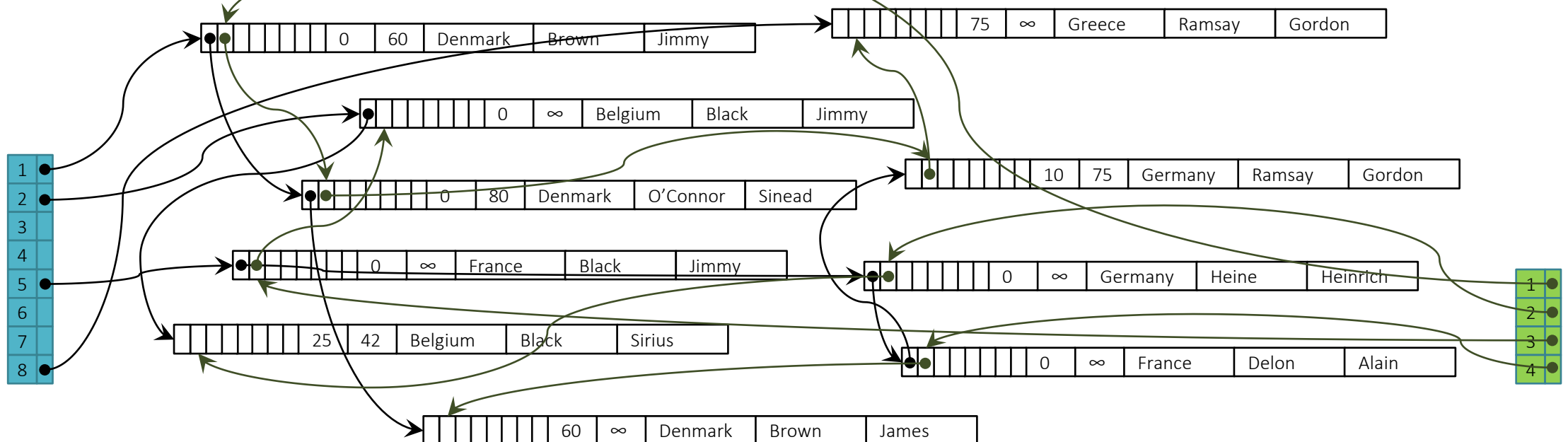
Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered hash index

Short name: Hash index

```
CREATE TABLE dbo.SampleMemOptTable
(
  CountryName varchar(30) NOT NULL,
  LastName varchar(60) NOT NULL,
  FirstName varchar(50) NOT NULL,
  INDEX ix_Hash_CountryName
    NONCLUSTERED HASH (CountryName)
    WITH (BUCKET_COUNT = 8),
  INDEX ix_Hash_LastName_FirstName
    NONCLUSTERED HASH (LastName, FirstName)
    WITH (BUCKET_COUNT = 4)
) WITH (MEMORY_OPTIMIZED = ON, DURABILITY = SCHEMA_AND_DATA);
```

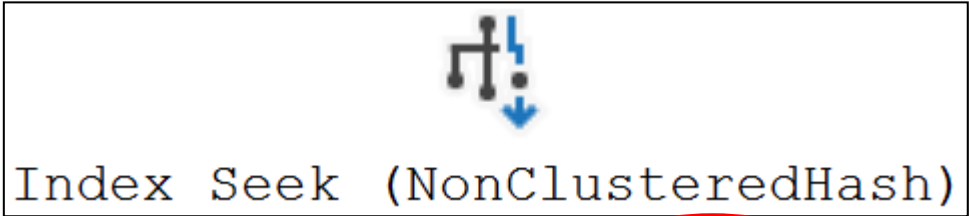


Memory-optimized indexes

Memory-optimized index structure

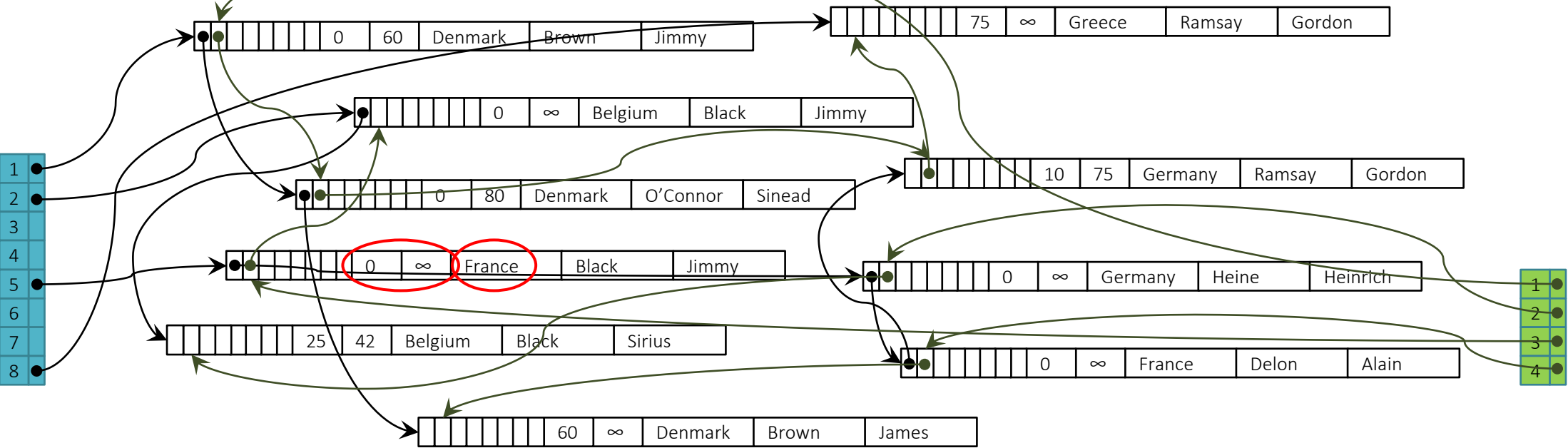
Memory-optimized nonclustered hash index

Short name: Hash index



Seek Predicate: Country = 'Germany'

Hash: 5



Memory-optimized indexes

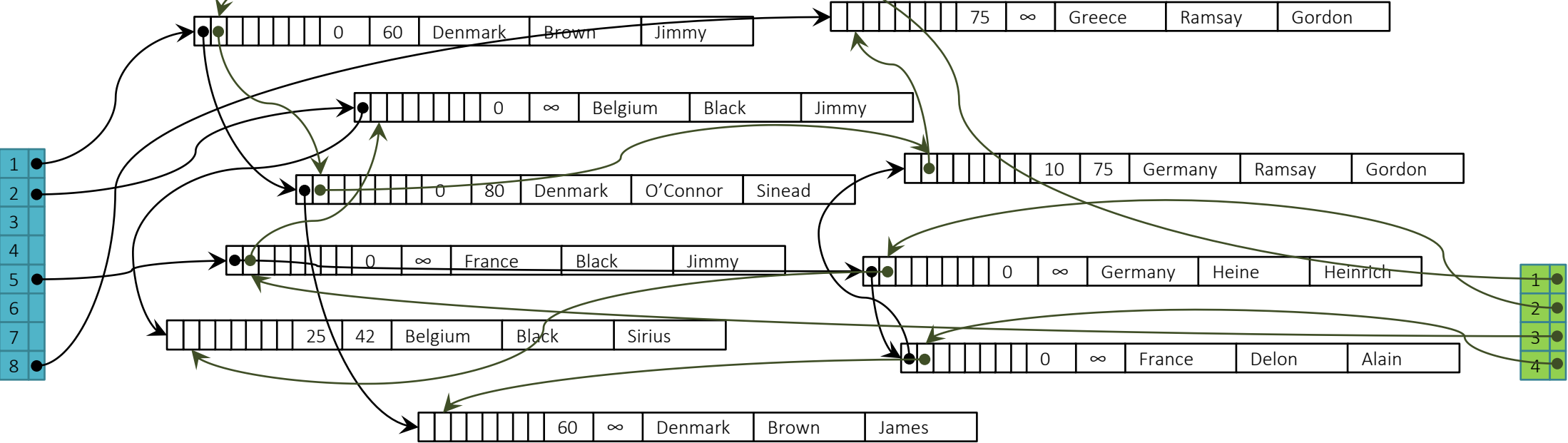


Index Scan (NonClusteredHash)

Memory-optimized index structure

Memory-optimized nonclustered hash index

Short name: Hash index



Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered hash index

Short name: Hash index

Index Scan operator is actually Table Scan in disguise

Why not scan each bucket in order?

- Order of buckets doesn't correspond to order of index values

- Data within a bucket fully unordered

- Might even be multiple index values in a single bucket

Index Scan would, effectively, be unordered

(Just like Table Scan)



Index Scan (NonClusteredHash)

Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered hash index

Short name: Hash index

Index Scan operator is actually Table Scan in disguise

Index Seek is very effective

Lots of buckets to reduce chance of hash collisions

Only equality search on full value

No inequality

No range search

No wildcard search



Index Seek (NonClusteredHash)

Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered hash index

Short name: Hash index

Index Scan operator is actually Table Scan in disguise

Index Seek is very effective

Lots of buckets to reduce chance of hash collisions

Only equality search on full value

Composite hash index:

Only search on **ALL** columns



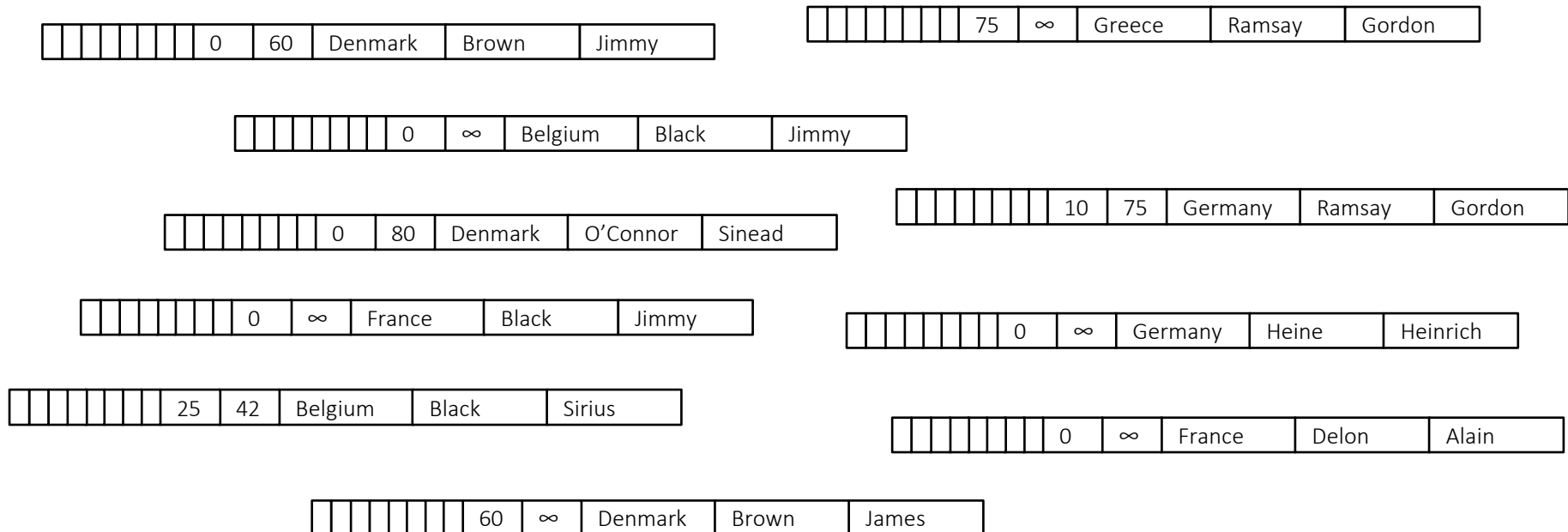
Index Seek (NonClusteredHash)

Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered index

Short name: Nonclustered index

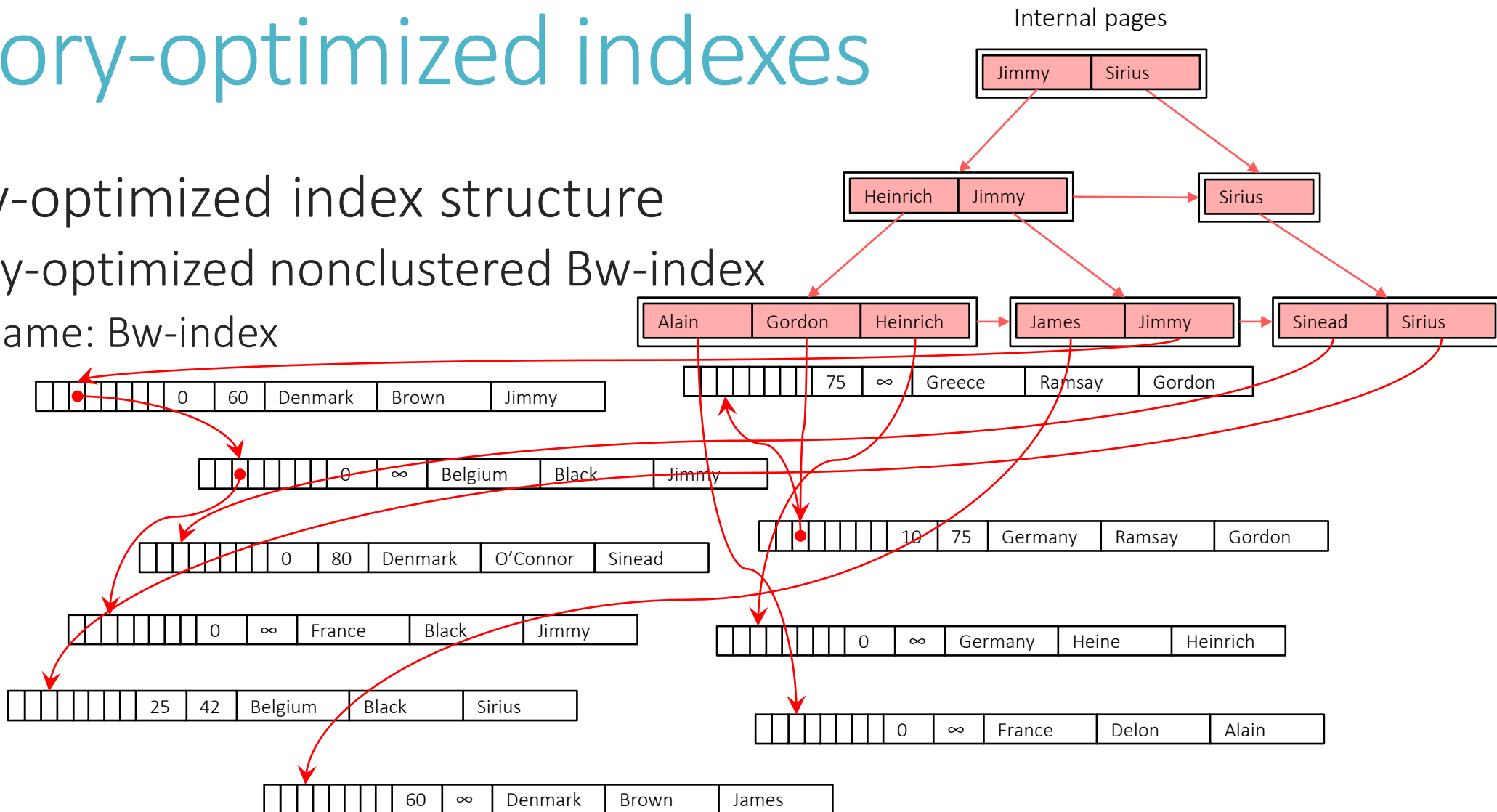


Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered Bw-index

Short name: Bw-index

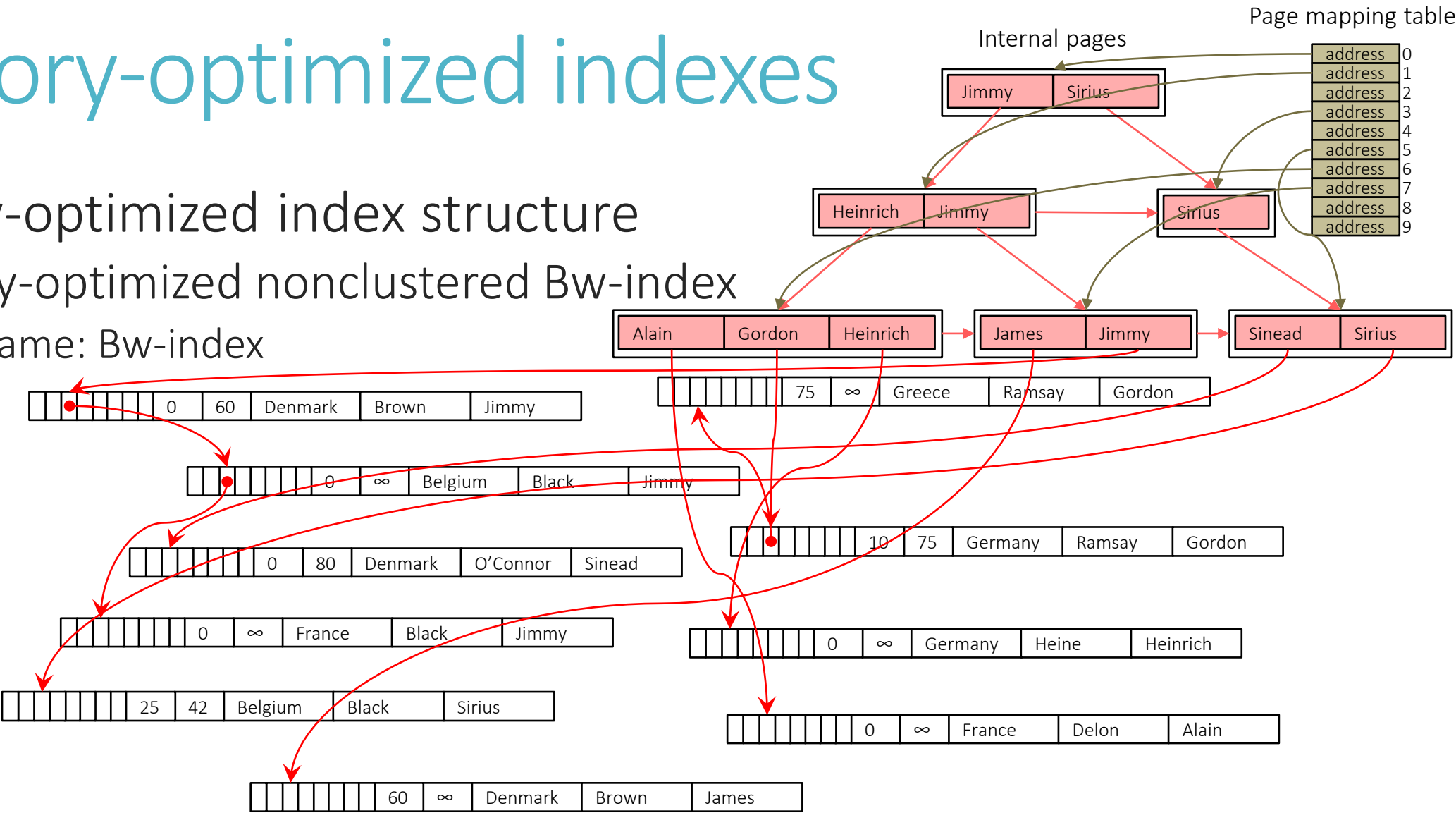


Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered Bw-index

Short name: Bw-index

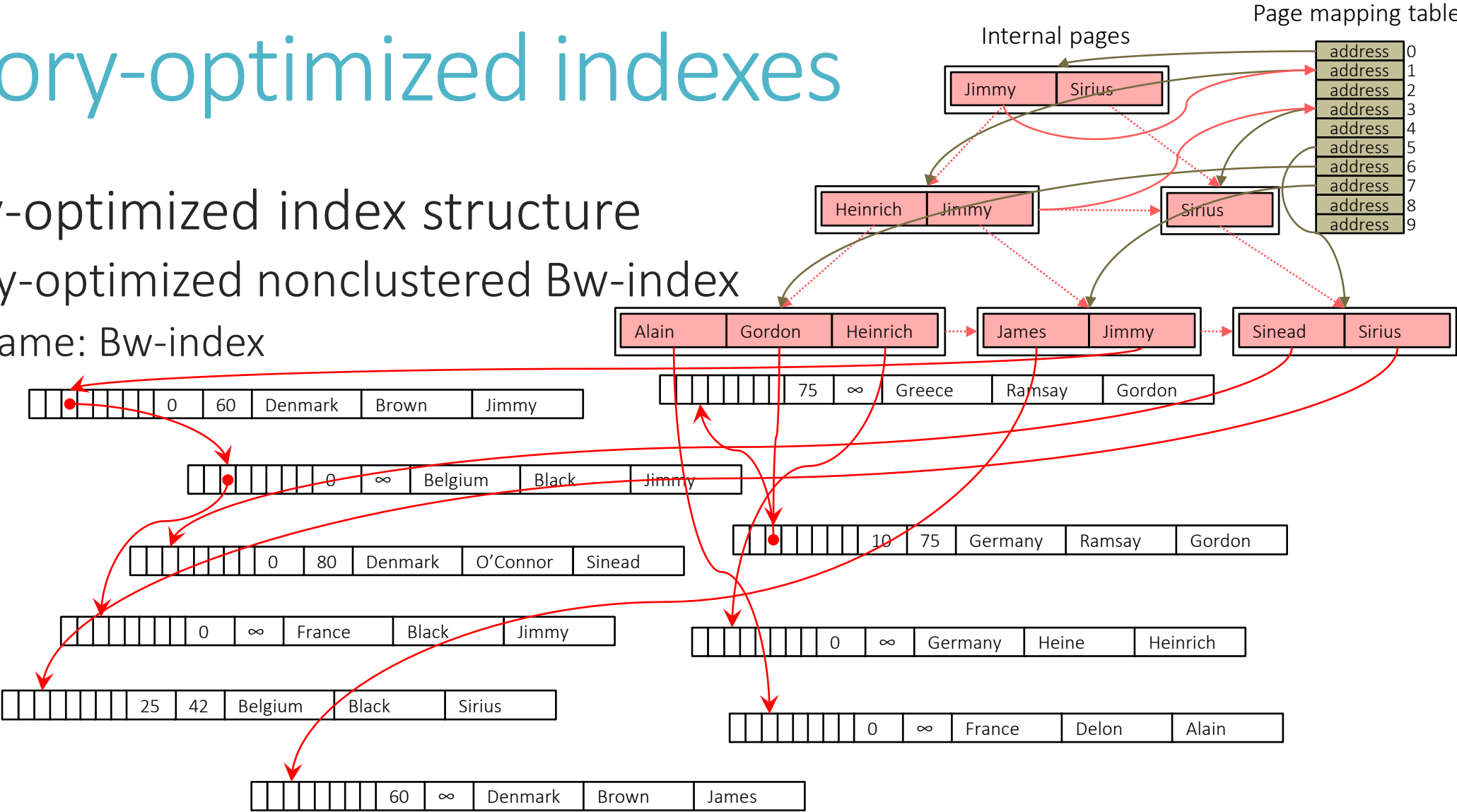


Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered Bw-index

Short name: Bw-index

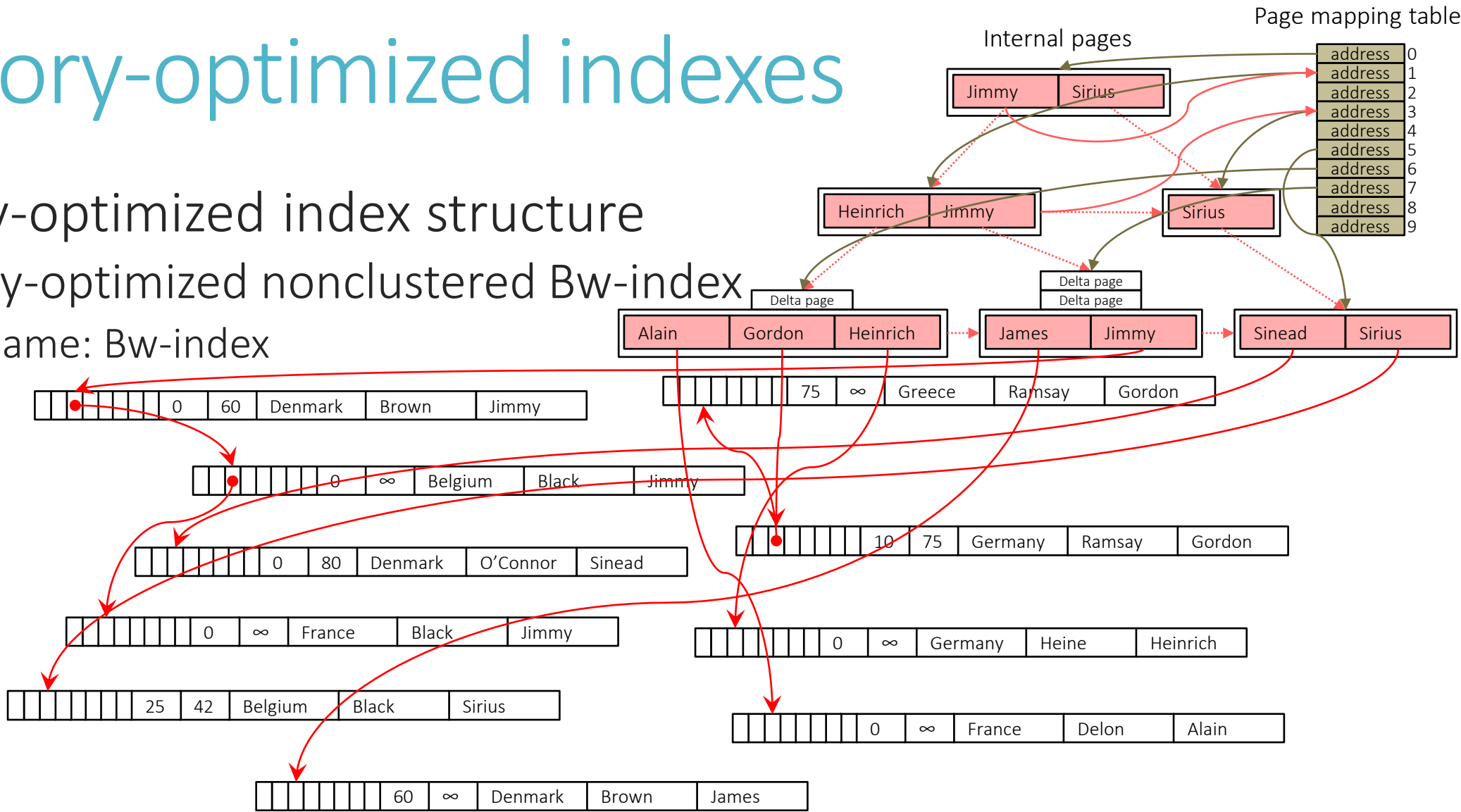


Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered Bw-index

Short name: Bw-index

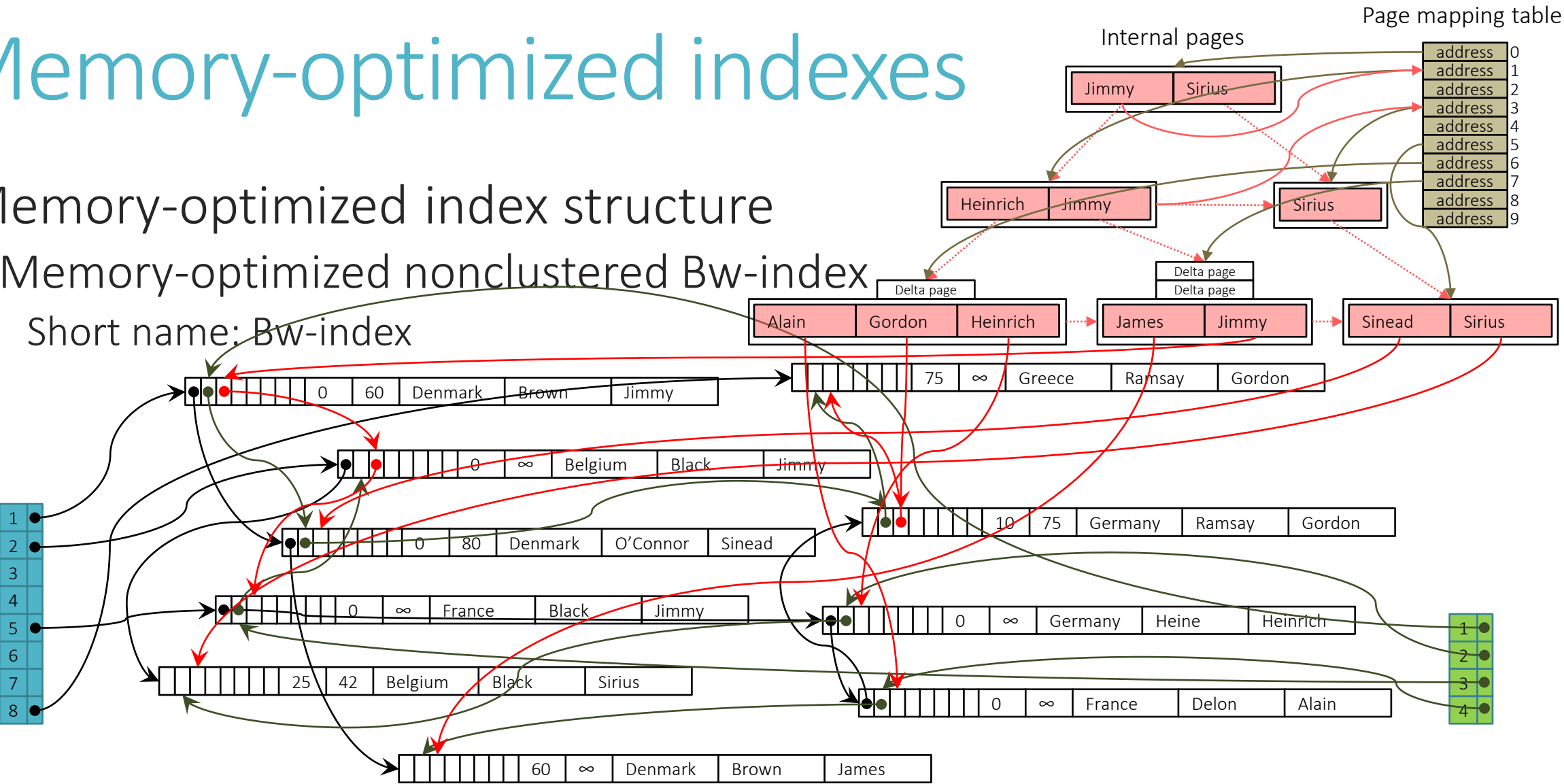


Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered Bw-index

Short name: Bw-index

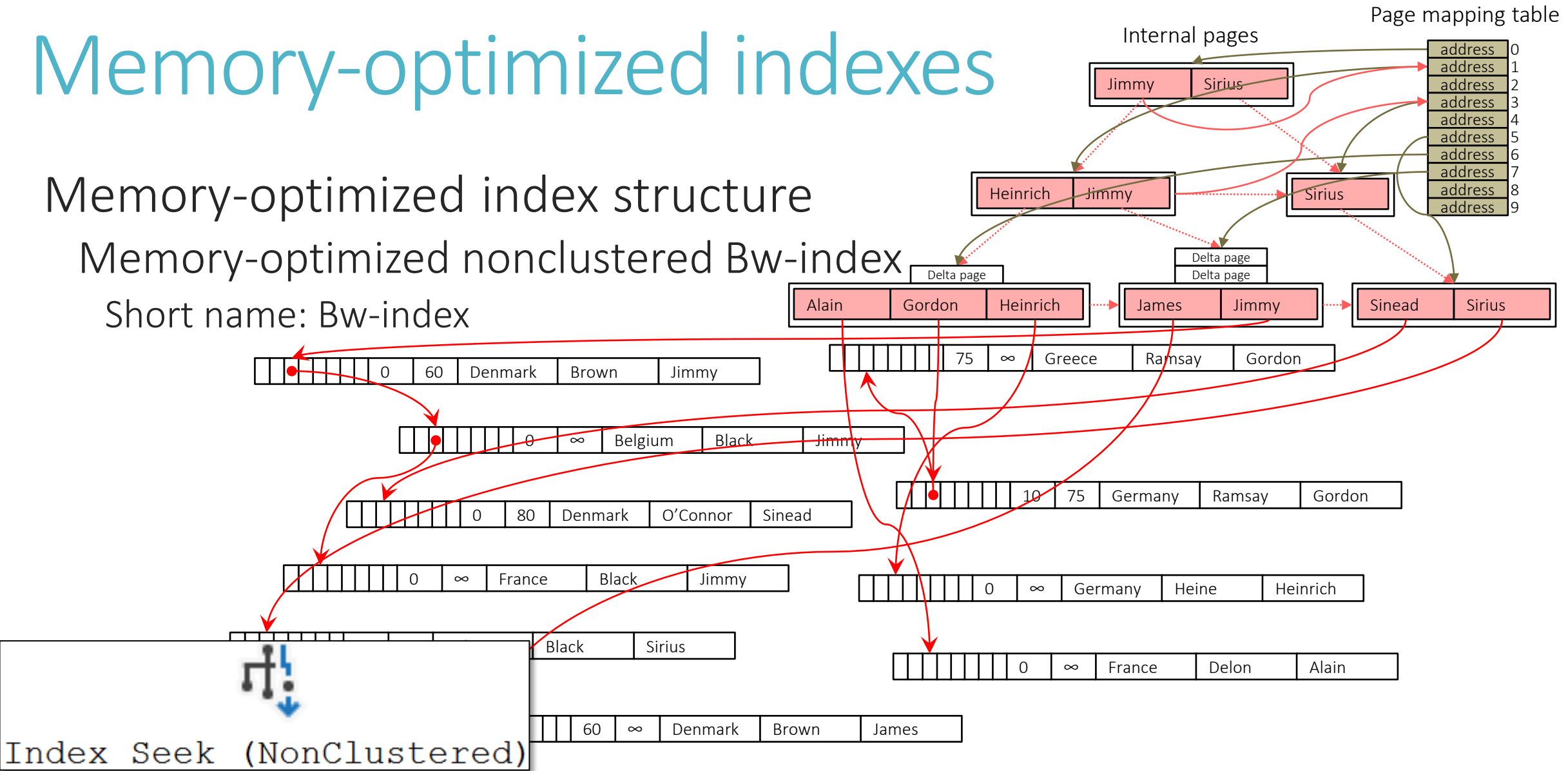


Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered Bw-index

Short name: Bw-index



Memory-optimized indexes

Memory-optimized index structure

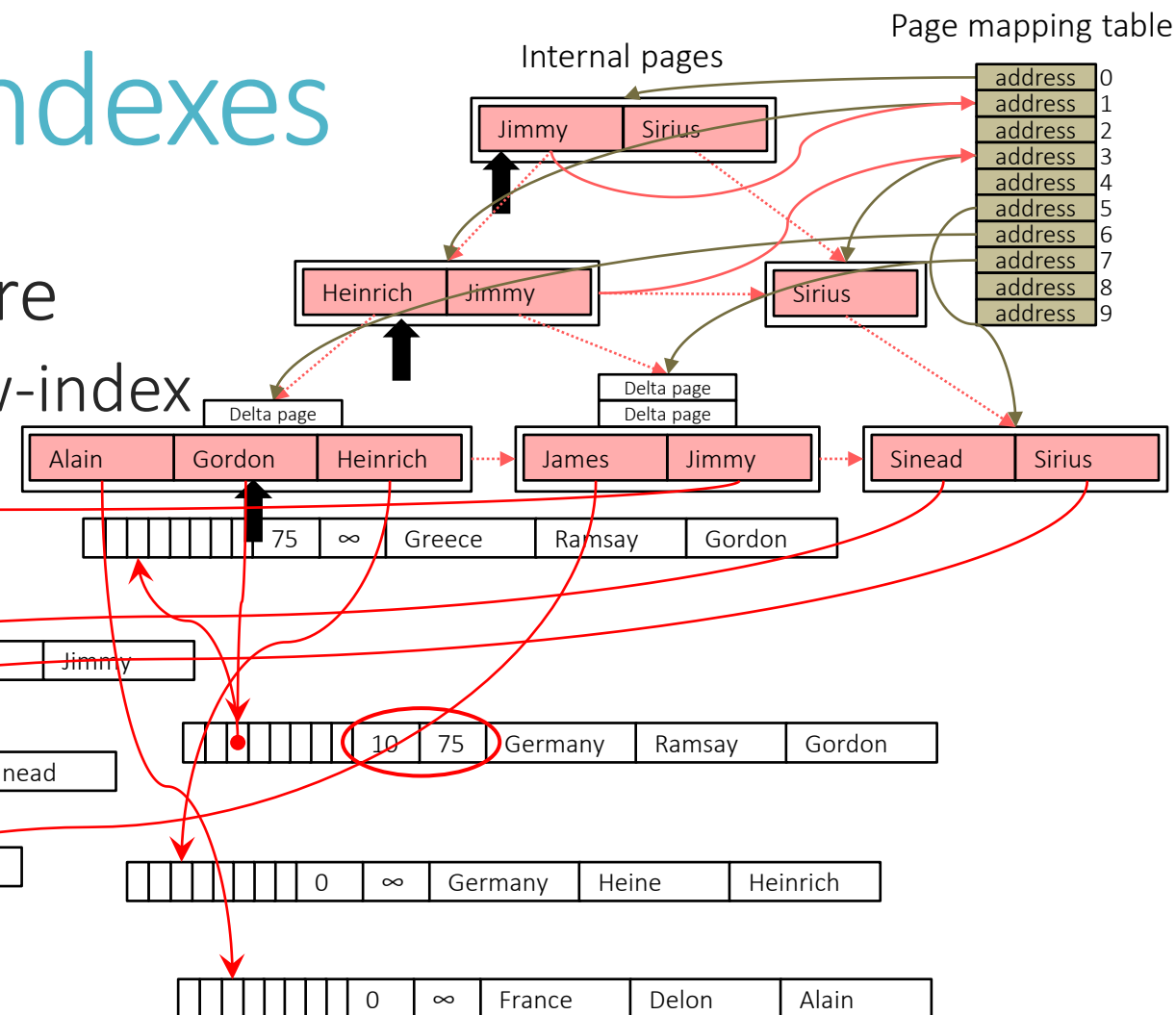
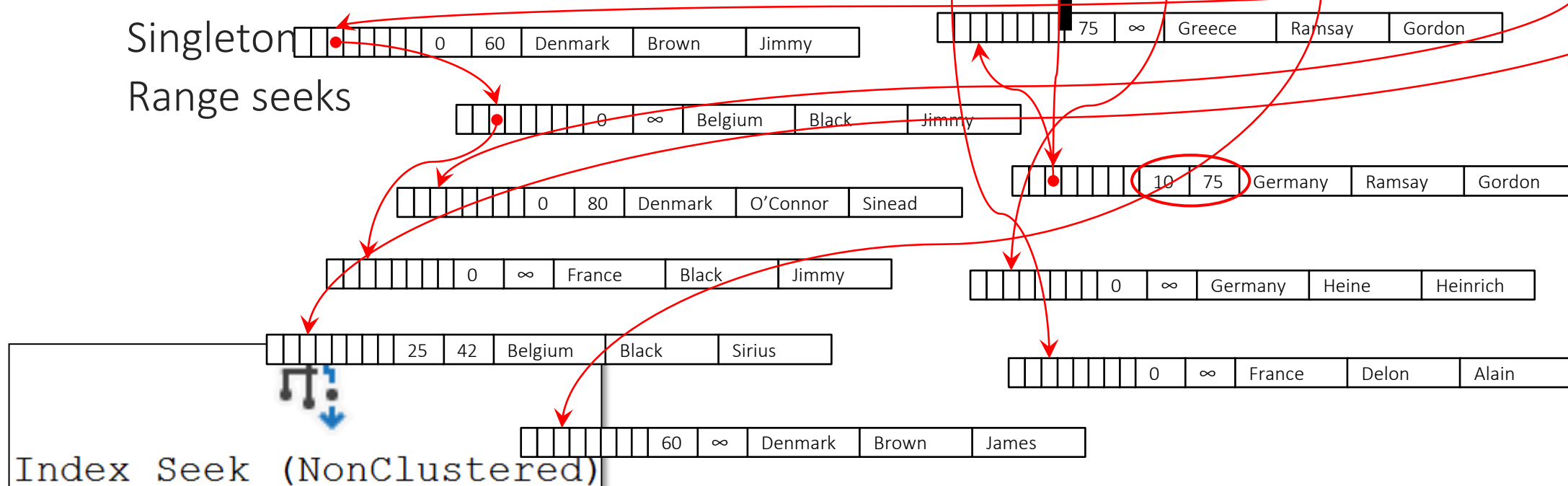
Memory-optimized nonclustered Bw-index

Short name: Bw-index

Singleton

					0	60	Denmark	Brown	Jimmy
--	--	--	--	--	---	----	---------	-------	-------

Range seeks



Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered Bw-index

Short name: Bw-index

Singleton

										0	60	Denmark	Brown	Jimmy
--	--	--	--	--	--	--	--	--	--	---	----	---------	-------	-------


Range seeks

										0	∞	Belgium	Black	Jimmy
--	--	--	--	--	--	--	--	--	--	---	---	---------	-------	-------

Ordered scan

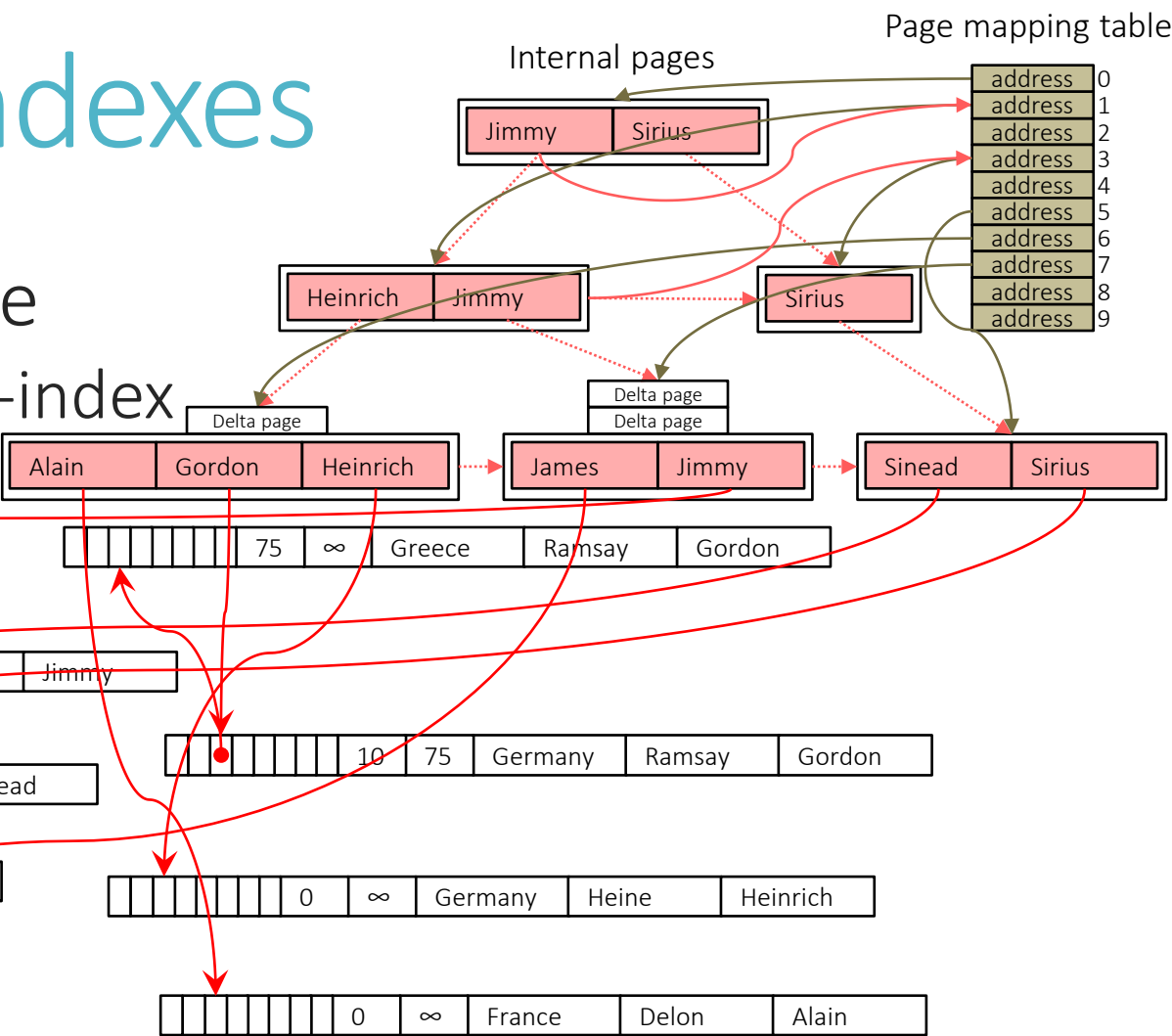
										0	80	Denmark	O'Connor	Sinead
--	--	--	--	--	--	--	--	--	--	---	----	---------	----------	--------

										0	∞	France	Black	Jimmy
--	--	--	--	--	--	--	--	--	--	---	---	--------	-------	-------



Index Scan (NonClustered)

													Black	Sirius
										60	∞	Denmark	Brown	James



Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered Bw-index

Short name: Bw-index

Singleton

						0	60	Denmark	Brown	Jimmy
--	--	--	--	--	--	---	----	---------	-------	-------

Range seeks

						0	∞	Belgium	Black	Jimmy
--	--	--	--	--	--	---	---	---------	-------	-------

Ordered scan

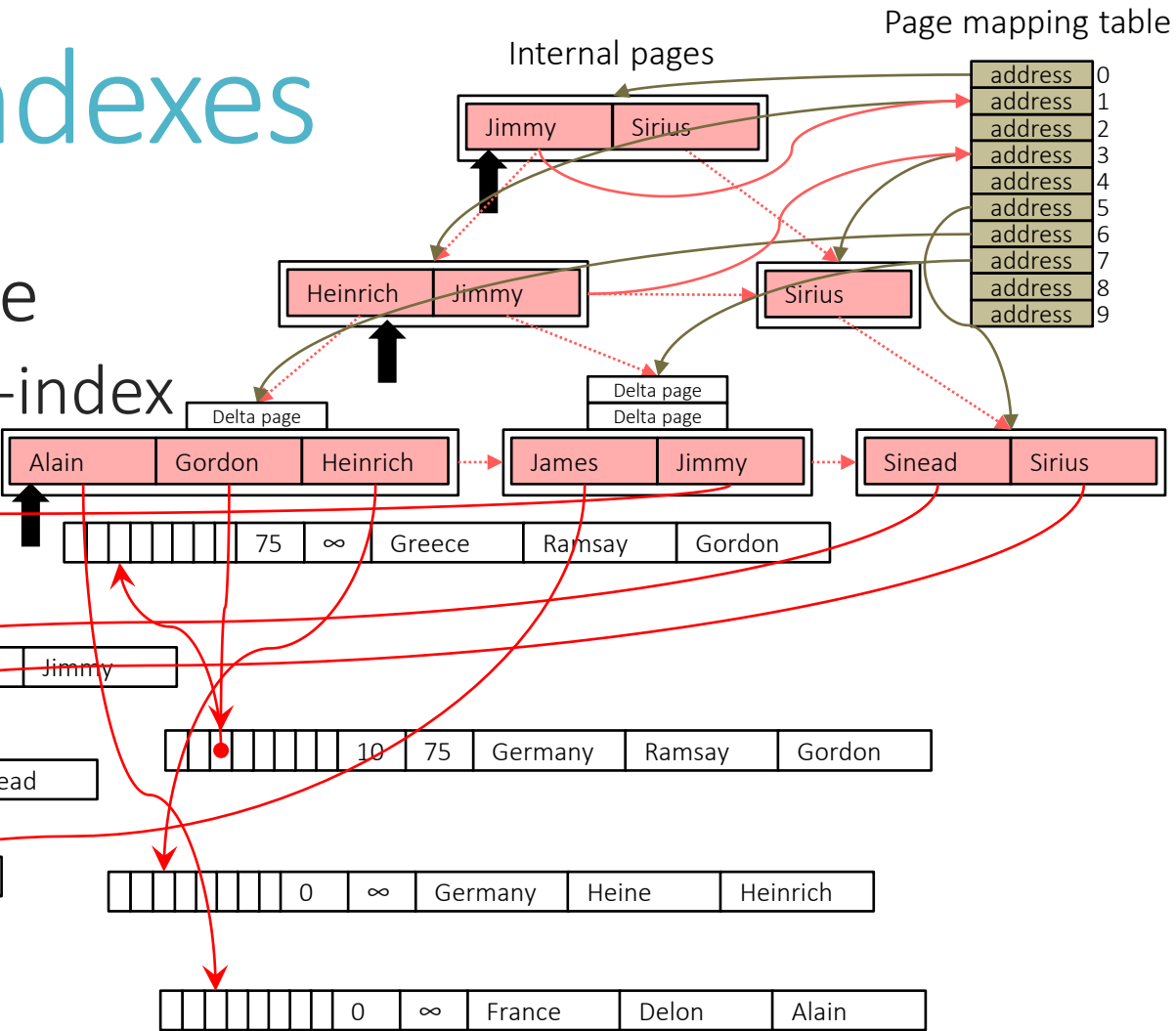
						0	80	Denmark	O'Connor	Sinead
--	--	--	--	--	--	---	----	---------	----------	--------

						0	∞	France	Black	Jimmy
--	--	--	--	--	--	---	---	--------	-------	-------

						25	42	Belgium	Black	Sirius
--	--	--	--	--	--	----	----	---------	-------	--------

						60	∞	Denmark	Brown	James
--	--	--	--	--	--	----	---	---------	-------	-------

Index Scan (NonClustered)



Memory-optimized indexes

Memory-optimized index structure

Memory-optimized nonclustered Bw-index

- Short name: Bw-index

- Singleton

- Range seeks

- Ordered scan

 - Always used, regardless of *Ordered* property

 - Scan Direction* property can only be FORWARD

Memory-optimized indexes

Memory-optimized index structure

Memory-optimized columnstore index

Introduced in SQL Server 2016

Structure mostly the same

But stored in memory, not on disk

Differences not in scope for this course

In-memory OLTP

Memory-optimized indexes

Natively compiled stored procedures

- Compiles stored procedure (with execution plan) in a DLL

 - Saves searching for existing plan in plan cache

 - Saves recompile cost if no existing plan found

 - Lots of restrictions

 - Can only read from and write to memory-optimized tables

 - Other restrictions: <https://tinyurl.com/NativeComp>

In-memory OLTP

Memory-optimized indexes

Natively compiled stored procedures

Compiles stored procedure (with execution plan) in a DLL

Supported operators:

- Index Seek

- Index Scan

- Table Scan

- Nested Loops

- Top

- Sort

- Compute Scalar

- Stream Aggregate

Summary

Memory-optimized indexes

Hash index

- Index Seek for equality on all columns

- No scan

Bw-index

- Supports range seeks, inequality, etc

- Ordered scan

Columnstore

Natively compiled stored procedure

- Always check performance gain or loss!

Next chapters

Chapter 3: Special index types

- Filtered index

- XML index

- Spatial index

- Full-text index

Chapter 4: Reading data in parallel or batch mode

Chapter 5: Assorted read optimizations