Physical Access

Indexes and Relation Layout

Yihao Sun Utah State University Sept. 2026

Theater Addresses

Which theaters show a movie, and what are their addresses?

$$\begin{aligned} \text{Movies}: &\quad M(\text{Title}, \text{Director}, \text{Actor}) \\ \text{Pariscope}: &\quad P(\text{Theater}, \text{Title}, \text{Schedule}) \\ \text{Location}: &\quad L(\text{Theater}, \text{Address}, \text{Phone}) \end{aligned}$$

$$\text{ans}(h, a) \leftarrow \underbrace{M(t, d, c),\ P(h, t, s)}_{M \bowtie P},\ L(h, a, p)$$

Today: how do we compute the first join?

One Step of the First Join

$$\text{ans}(h, a) \leftarrow \underbrace{M(t, d, c),\ P(h, t, s)}_{M \bowtie P},\ L(h, a, p)$$

Choose one tuple from Movies.

$u = \langle \text{Title} : \text{filmA},\ \text{Director} : \text{Bergman},\ \text{Actor} : \text{actor1} \rangle \in M$

$\sigma_{\text{Title}=u(\text{Title})}(P) = \sigma_{\text{Title}=\text{filmA}}(P)$

Theater	Title	Schedule
Champo	filmA	18:00
Champo	filmA	21:00
Rialto	filmB	20:00

Which Pariscope tuples share this Title?

Scanning for Matches

Find all showings for filmA.

Theater	Title	Schedule	Matches?
Champo	filmA	18:00	yes
Champo	filmA	21:00	yes
Rialto	filmB	20:00	no

Three Title tests. Two matching tuples.

The next movie also asks for filmA. Scan again?

An Index Organizes Matches

$$\sigma_{\text{Title}=a}(P) = \{v \in P \mid v(\text{Title}) = a\}$$

Search key → **Matching tuples**

filmA

Theater	Title	Schedule
Champo	filmA	18:00
Champo	filmA	21:00

filmB

Theater	Title	Schedule
Rialto	filmB	20:00

One key can identify several matches—or none.

Lookup and Fetch

Index on Title

Title	Row IDs
filmA	0, 1
filmB	2

Stored Pariscope rows

Row ID	Theater	Title	Schedule
0	Champo	filmA	18:00
1	Champo	filmA	21:00
2	Rialto	filmB	20:00

$\sigma_{\text{Title=filmA}}(P)$

Lookup finds locations. Fetch recovers tuples.

Sorted Array

Larger catalog: 12 rows, 8 titles

Lookup filmF

filmD $\longrightarrow$ filmF $\longrightarrow$ $\{4,11\}$

0	1	2	3	4	5	6	7
filmA	filmB	filmC	filmD	filmE	**filmF**	filmG	filmH
1, 5, 10	6	3	0, 8	9	**4, 11**	7	2

filmF $\leq$ Title $\leq$ filmG

$\{4,11\} \cup \{7\}$

Locate the lower bound; read through filmG; stop at filmH.

New keys may require moving array entries.

Hash Table

Larger catalog: 12 rows, 8 titles

Lookup filmF

$$A = 0,\ldots,H = 7; \quad h(k) = \text{num}(k) \bmod 4$$



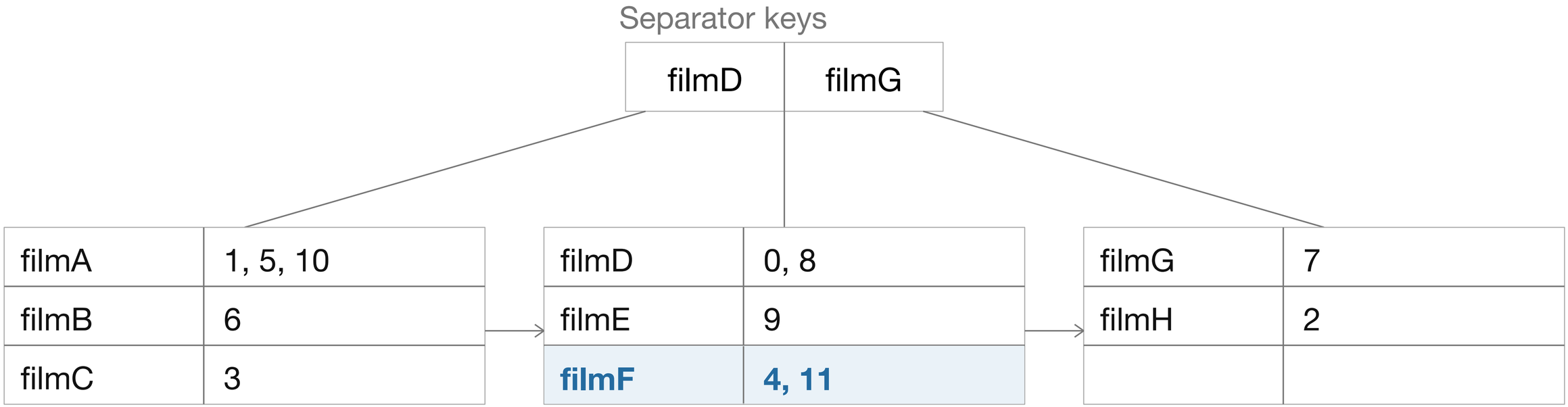
$h(\text{filmF}) = 1; \quad \text{filmB} \neq \text{filmF}; \quad \text{filmF} \mapsto \{4,11\}$

A shared bucket still requires a key comparison.

B+ Tree

Larger catalog: 12 rows, 8 titles

$\text{Title} = \text{filmF} \longrightarrow \{4,11\}$



Leaves: title and matching row IDs

filmF…filmG : continue along the leaf link

Choosing an Access Structure

Every access returns row IDs 4 and 11 for filmF.

Equality, ranges, or frequent changes?

Access	Equality	Ordered range	When data changes
Scan	Test every row	Test every row	No index to maintain
Hash	Bucket; check key	No ordered path	Grow or resize
Sorted array	Binary search	Locate; scan slice	May shift entries
B+ tree	Root to leaf	Locate; follow leaves	May split nodes

Also count matching entries, row fetches, and maintenance.

Indexed Join

for each $u \in M$:

  for each $v \in \sigma_{\text{Title}=u(\text{Title})}(P)$ using the Title index :

    emit $u \bowtie v$

Movie (Title, Director, Actor)	Lookup key	Inner row IDs	Join tuples
filmA, Bergman, actor1	filmA	0, 1	2
filmA, Bergman, actor2	filmA	0, 1	2
filmB, Other, actor3	filmB	2	1

Three outer tuples. Five join tuples.

Repeated keys can belong to different tuples.

Which Values Must We Read?

At what times is filmA showing?

$$\text{ans}(s) \leftarrow P(h, \text{filmA}, s)$$

$$\pi_{\text{Schedule}}\left(\sigma_{\text{Title=filmA}}(P)\right)$$

Theater	Title	Schedule
Champo	filmA	18:00
Champo	filmA	21:00
Rialto	filmB	20:00

Which columns test the condition? Which column supplies the answer?

Put Title Values Together

$$\pi_{\text{Schedule}}\left(\sigma_{\text{Title=filmA}}(P)\right)$$

N-ary Storage Model (NSM)

Champo	filmA	18:00	Champo	filmA	21:00	Rialto	filmB	20:00

Decomposed Storage Model (DSM)

Champo	Champo	Rialto	filmA	filmA	filmB	18:00	21:00	20:00

1. Scan Title **2. Fetch Schedule**

Inter-record locality: one attribute's values stored together.
Toy model: 3 equal-size fields per cache transfer.
Rows: 3 transfers. Columns: 2, including Schedule.

Which Original Row Survived?

Filtering changes positions. How do we keep row identity?

$$\begin{aligned} \text{Title}: &\quad (0,\text{filmA}),\ (1,\text{filmA}),\ (2,\text{filmB}) \\ \text{Schedule}: &\quad (0,18{:}00),\ (1,21{:}00),\ (2,20{:}00) \end{aligned}$$

$$\begin{aligned} \text{Title} = \text{filmB}: &\quad \text{source row } 2 \longrightarrow \text{filtered position } 0 \\ \text{fetch source row } 2: &\quad 20{:}00 \quad \text{correct} \\ \text{fetch filtered position } 0: &\quad 18{:}00 \quad \text{wrong} \end{aligned}$$

Decomposition Storage Model (DSM): pair values with row IDs.
Aligned columns can keep those IDs implicit.
Row IDs link values; layout decides how close they are.

Fetch the Next Join Tuple

Rows (NSM) $u(\text{Title}) = \text{filmA}$ $\xrightarrow{\text{Title index on } P}$ row IDs $0,1$

P0			P1			P2		
Champo	filmA	18:00	Champo	filmA	21:00	Rialto	filmB	20:00

Columns

Theater			Title (known)			Schedule		
Champo	Champo	Rialto	filmA	filmA	filmB	18:00	21:00	20:00
0	1	2	0	1	2	0	1	2

Fetch Theater and Schedule for row 1; Title is already known.
Intra-record locality: a tuple's different attributes stored together.
Several matches may reuse the same column blocks.

Columns Within a Page: PAX

PAX page

	Attribute minipages		
	0	1	2
Theater	Champo	Champo	Rialto
Title	filmA	filmA	filmB
Schedule	18:00	21:00	20:00

Across tuples: column groups improve cache locality.
Within a tuple: its fields remain on the same page.
The page remains the I/O unit.

DuckDB: Column Data in Row Groups

Storage

Row group

Theater	Title	Schedule
Champo	filmA	18:00
Champo	filmA	21:00
Rialto	filmB	20:00

Column segments
backed by disk blocks

The same layout ideas

Read needed columns

Compress each column

Keep row positions aligned

Which values will the query need next—and are they stored together?