ECS 165: Database Systems

Project's First Milestone

Winter 2025

UCDAVIS
UNIVERSITY OF CALIFORNIA

Motivation

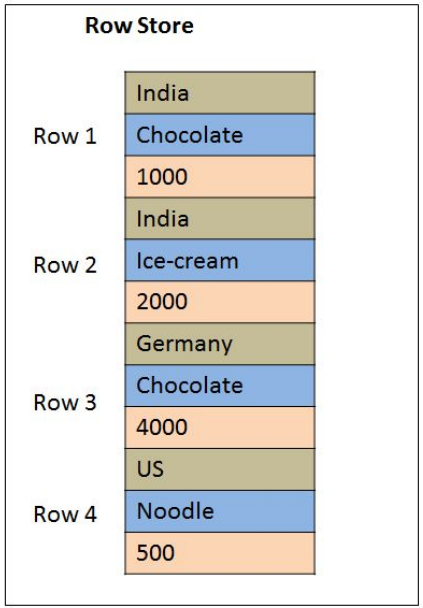
- Gap between managing high velocity **Updates** and **Reads**
- **OLAP** (Online Analytical Processing) - Read-Intensive
- **OLTP** (Online Transaction Processing) - Update-Intensive

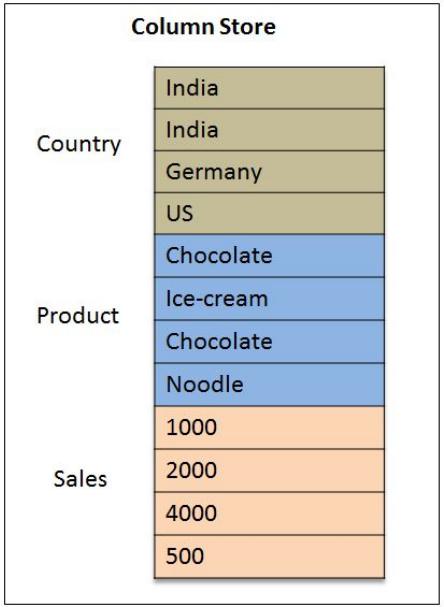
- **Lineage-Store (L-Store)**

Background

- Row Store Database
  - Different columns are stored in separate columns
  - Update-friendly
  - Reading unwanted data
- Column Store Database
  - Different columns are stored in the same page
  - Read-Friendly
  - Costly updates

	Country	Product	Sales
Row 1	India	Chocolate	1000
Row 2	India	Ice-cream	2000
Row 3	Germany	Chocolate	4000
Row 4	US	Noodle	500





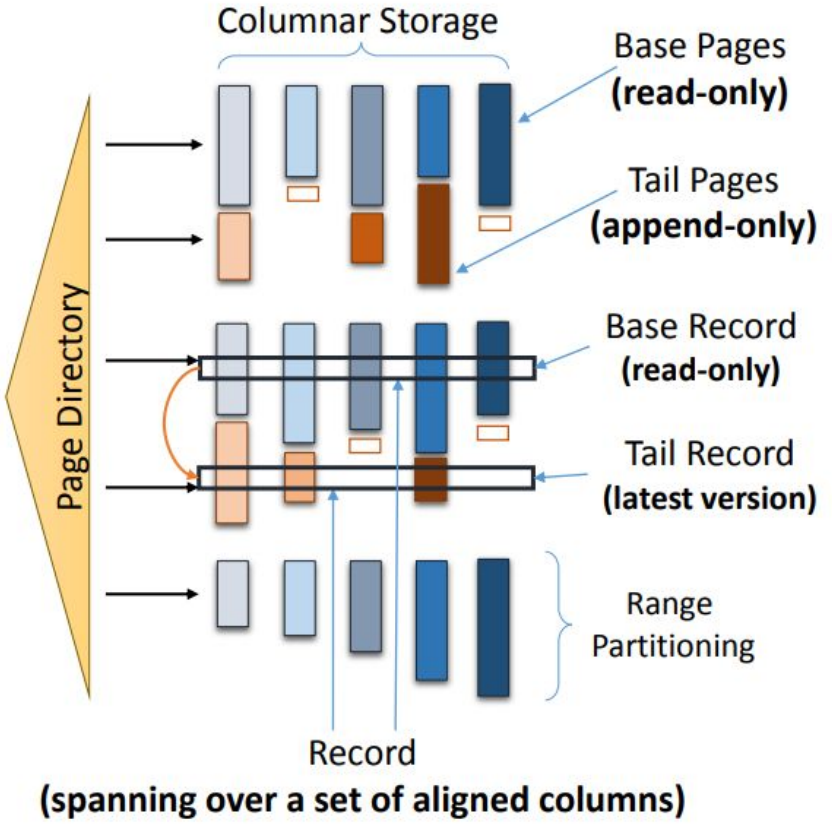
UCDAVIS

Lineage-Store (L-Store)

- L-Store is Column Store
  - Avoid reading Irrelevant Data
  - Improve Data Homogeneity
  - Better Compression ratio
- What About Updates?

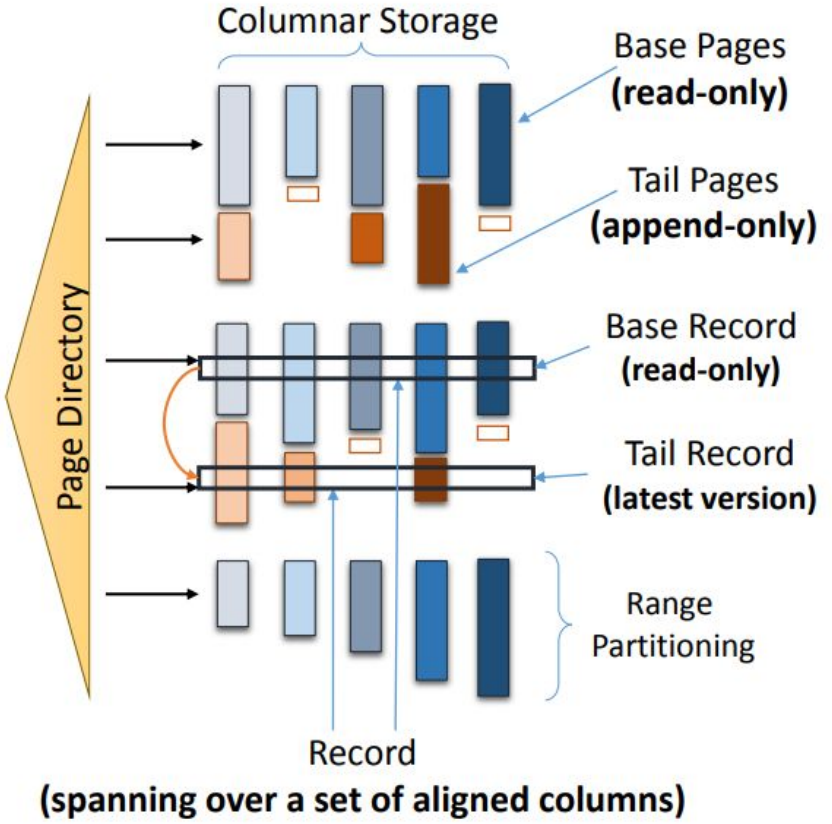
L-Store Architecture

- Virtually Disjoint Ranges
- ReadOnly Base Pages
- Append Only Tail Pages
  - Latest Updates
  - New record for each update



L-Store Architecture

- Insert: Base Page
- Update: Tail Page
- Read: Base and Tail Page
- Multi Version
  - Linkage Between versions
- Merge: Contention Free



L-Store in Depth

- Each record has an RID
- PageDirectory: RID ⇒ physical location
- **Index:** key ⇒ RID of base record (original value)
- MetaData Columns
  - **Indirection**
  - Schema Encoding
  - Start Time
  - Last Update

L-Store in Depth

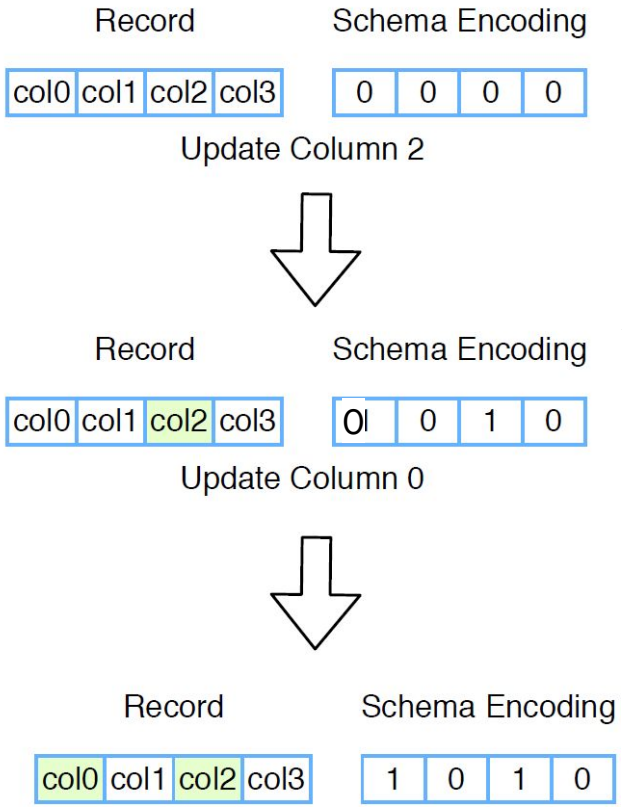
Indirection Column

- The Indirection of Base Record stores the RID of the latest Tail Record (Base record points to the latest tail record)
- Quick Access To the latest version
- The Indirection of Tails Record stores the RID of the previous Tail Record (Each tail record points to previous one)
- Cumulative vs Non-Cumulative updates

L-Store in Depth

Schema Encoding Column

- Bitmap Representation of the state
- In base record it keeps track of updated columns
- Non-Cumulative updates
  - In Tail Records specifies the valid columns



Example

https://arxiv.org/abs/1601.04084

RID	*Indirection*	*Schema Encoding*	*Start Time*	*Key*	*A*	*B*	*C*
Partitioned base records for the key range of k_1 to k_3							
b_1	t_8	0000	10:02	k_1	a_1	b_1	c_1
b_2	t_5	0101	13:04	k_2	a_2	b_2	c_2
b_3	t_7	0001	15:05	k_3	a_3	b_3	c_3
Partitioned base records for the key range of k_4 to k_6							
b_4	$\perp$	0000	16:20	k_4	a_4	b_4	c_4
b_5	$\perp$	0000	17:21	k_5	a_5	b_5	c_5
b_6	$\perp$	0000	18:02	k_6	a_6	b_6	c_6
Partitioned tail records for the key range of k_1 to k_3							
t_1	b_2	0100*	13:04	$\varnothing$	a_2	$\varnothing$	$\varnothing$
t_2	t_1	0100	19:21	$\varnothing$	a_{21}	$\varnothing$	$\varnothing$
t_3	t_2	0100	19:24	$\varnothing$	a_{22}	$\varnothing$	$\varnothing$
t_4	t_3	0001*	13:04	$\varnothing$	$\varnothing$	$\varnothing$	c_2
t_5	t_4	0101	19:25	$\varnothing$	a_{22}	$\varnothing$	c_{21}
t_6	b_3	0001*	15:05	$\varnothing$	$\varnothing$	$\varnothing$	c_3
t_7	t_6	0001	19:45	$\varnothing$	$\varnothing$	$\varnothing$	c_{31}
t_8	b_1	0000	20:15	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

UCDAVIS

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encoding)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0000	BID1

Make two updates:

- C1 ⇒ C2
- B1 ⇒ B2

Make two updates:

- C1 ⇒ C2
- B1 ⇒ B2

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
TID1	k1	A1	B1	C2	0001	BID1
TID2	k1	A1	B2	C2	0011	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage0, TailPage1, ... TailPage5), offset = 1	
TID2	pages = (TailPage0, TailPage1, ... TailPage5), offset = 2	

Make two updates:

- C1 ⇒ C2
- B1 ⇒ B2

What is the problem of such an implementation?

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
TID1	k1	A1	B1	C2	0001	BID1
TID2	k1	A1	B2	C2	0011	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage0, TailPage1, ... TailPage5), offset = 1	
TID2	pages = (TailPage0, TailPage1, ... TailPage5), offset = 2	

Make two updates:

- C1 ⇒ C2
- B1 ⇒ B2

What is the problem of such an implementation?

Waste time and space for writes to pages of unaffected columns.

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
TID1	k1	A1	B1	C2	0001	BID1
TID2	k1	A1	B2	C2	0011	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage0, TailPage1, ... TailPage5), offset = 1	
TID2	pages = (TailPage0, TailPage1, ... TailPage5), offset = 2	

Make two updates:

- C1 ⇒ C2
- B1 ⇒ B2

Non-Cumulative:

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
TID1				C2	0001	BID1
TID2			B2		0010	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage3, TailPage4, TailPage5), offset = 1	
TID2	pages = (TailPage2, TailPage4, TailPage5), offset = 2	

Make two updates:

- C1 ⇒ C2
- B1 ⇒ B2

Cumulative:

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
TID1				C2	0001	BID1
TID2			B2	C2	0011	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage3, TailPage4, TailPage5), offset = 1	
TID2	pages = (TailPage2, TailPage3, TailPage4, TailPage5), offset = 2	

VIS

Non-Cumulative vs Cumulative

- C1 ⇒ C2
- B1 ⇒ B2
- Read all updated values of the record

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
TID1				C2	0001	BID1
TID2			B2		0010	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage3, TailPage4, TailPage5), offset = 1	
TID2	pages = (TailPage2, TailPage4, TailPage5), offset = 2	

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
TID1				C2	0001	BID1
TID2			B2	C2	0011	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage3, TailPage4, TailPage5), offset = 1	
TID2	pages = (TailPage2, TailPage3, TailPage4, TailPage5), offset = 2	

What is the problem, for both versions?

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
TID1				C2	0001	BID1
TID2			B2		0010	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage3, TailPage4, TailPage5), offset = 1	
TID2	pages = (TailPage2, TailPage4, TailPage5), offset = 2	

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
TID1				C2	0001	BID1
TID2			B2	C2	0011	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage3, TailPage4, TailPage5), offset = 1	
TID2	pages = (TailPage2, TailPage3, TailPage4, TailPage5), offset = 2	

What is the problem, for both version?

We waste the ***red*** slots.

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
TID1				C2	0001	BID1
TID2			B2		0010	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage3, TailPage4, TailPage5), offset = 1	
TID2	pages = (TailPage2, TailPage4, TailPage5), offset = 2	

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
TID1				C2	0001	BID1
TID2			B2	C2	0011	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage3, TailPage4, TailPage5), offset = 1	
TID2	pages = (TailPage2, TailPage3, TailPage4, TailPage5), offset = 2	

Solution

RID	BasePage0 (key)	BasePage1 (A)	BasePage2 (B)	BasePage3 (C)	BasePage4 (Schema Encodeing)	BasePage5 (Indirectory)
BID1	k1	A1	B1	C1	0011	TID2

RID	TailPage0 (key)	TailPage1 (A)	TailPage2 (B)	TailPage3 (C)	TailPage4 (Schema Encodeing)	TailPage5 (Indirectory)
			B2	C2	0001	BID1
					0010	TID1

Page Directory

RID	Physical Locations	
BID1	pages = (BasePage0, BasePage1, ... BasePage5), offset = 1	
TID1	pages = (TailPage3, TailPage4, TailPage5), offset = 1	
TID2	***pages = (TailPage4, TailPage5), offset = 2; pages = (TailPage2), offset = 1;***	

UCDAVIS