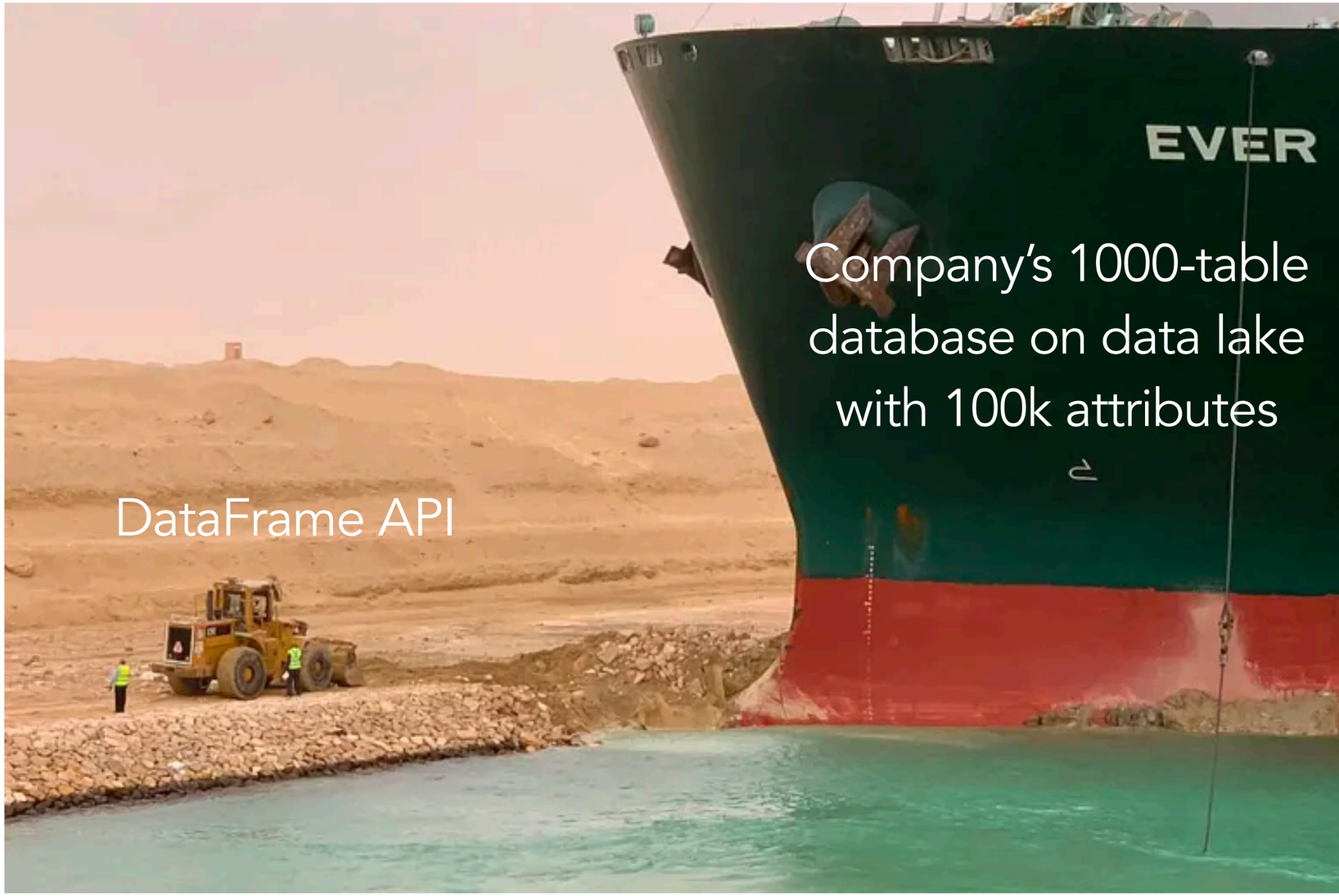
DSC 204a Scalable Data Systems

- Haojian Jin



Meme idea credit: https://datasystemsfun.tumblr.com/

Where are we in the class?

Foundations of Data Systems (2 weeks)

- Digital representation of Data → Computer Organization → Memory hierarchy → Process → Storage

Scaling Distributed Systems (3 weeks)

- Cloud → Network → **Distributed storage** → Partition and replication (HDFS) → Distributed computation

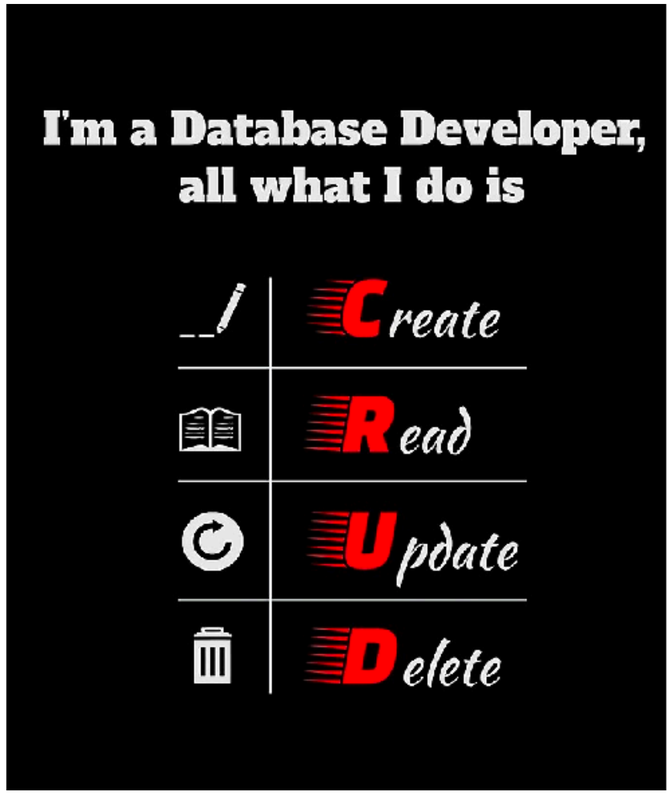
Data Processing and Programming model (5 weeks)

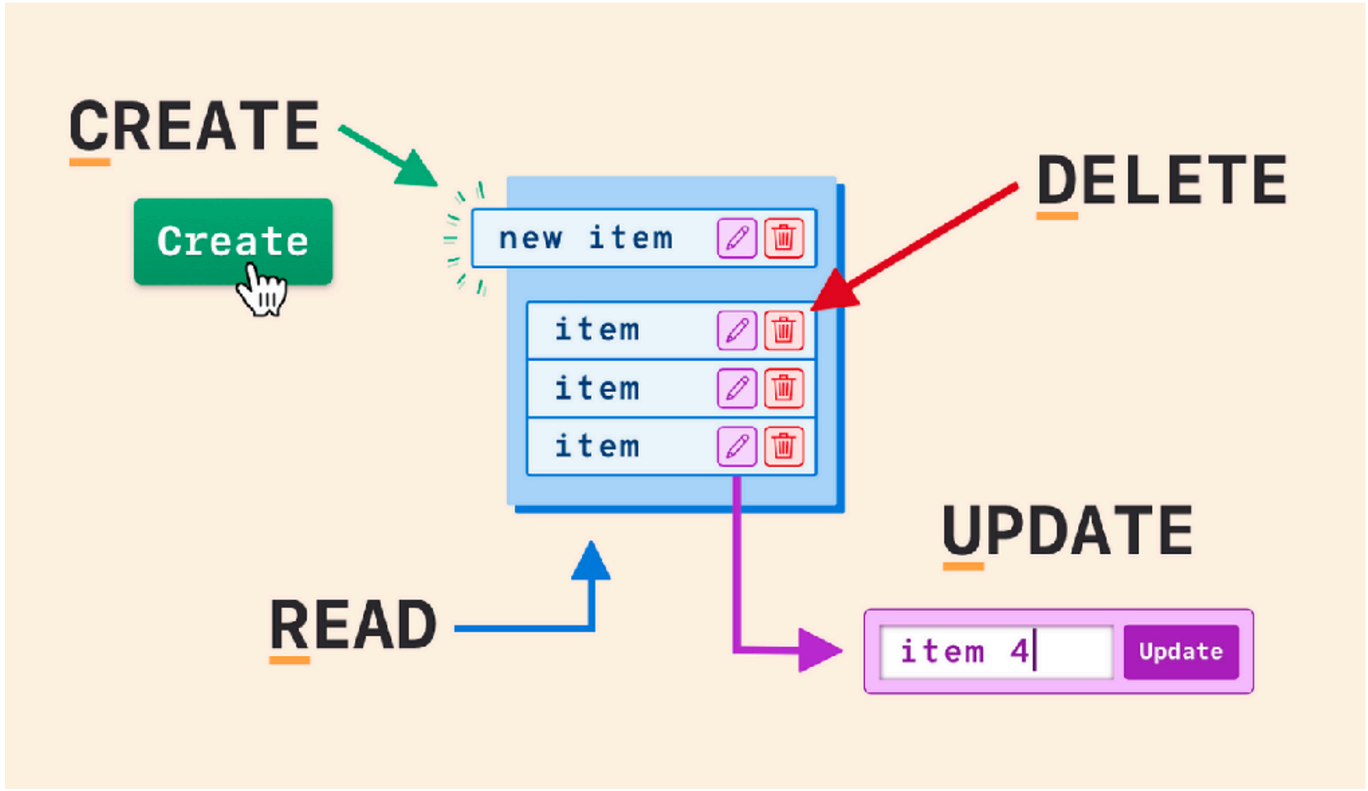
- Data Models evolution → Data encoding evolution → → IO & Unix Pipes → Batch processing (MapReduce) → Stream processing (Spark)

Today's topic: Column-oriented storage

- OLTP v.s. OLAP
- Data warehousing
- Schemas for Analytics
- Column-oriented storage
- Data cubes and materialized views

CRUD


I'm a Database Developer,
all what I do is
Create
Read
Update
Delete


CREATE
Create
new item
item
item
item
DELETE
UPDATE
READ
item 4
Update

Database transactions

- Make sale
- Place an order
- Pay an employee’s salary
- Comment a blog post
- Act in games
- Add/remove contract to an address book

Online transaction processing (OLTP)

Walmart Beer and Diaper (1988)

Forbes 1988

- Unexpected correlation:
  - Sales of diapers and beer

https://www.forbes.com/forbes/1998/0406/6107128a.html?sh=2574a9316260

Data analytics

- What was the total revenue of each of our stores in Jan?
- How many more bananas that usual did we sell during our latest data?
- Which brand of baby food is most often purchased together with brand X diapers?

Online analytic processing (OLAP)

OLTP v.s. OLAP

Property	Transaction processing systems (OLTP)	Analytic systems (OLAP)
Main read pattern	Small number of records per query, fetched by key	Aggregate over large number of records

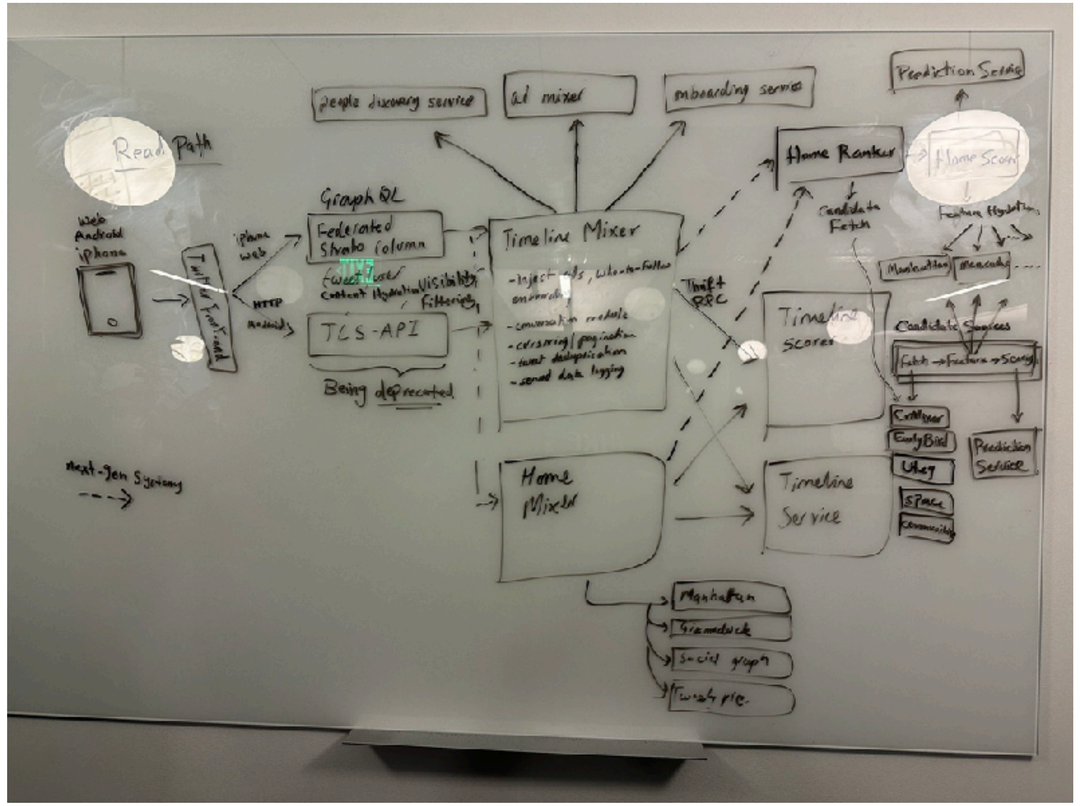
OLTP v.s. OLAP

Property	Transaction processing systems (OLTP)	Analytic systems (OLAP)
Main read pattern	Small number of records per query, fetched by key	Aggregate over large number of records
Main write pattern	Random-access, low-latency writes from user input	Bulk import (ETL) or event stream
Primarily used by	End user/customer, via web application	Internal analyst, for decision support
What data represents	Latest state of data (current point in time)	History of events that happened over time
Dataset size	Gigabytes to terabytes	Terabytes to petabytes

Today's topic: Column-oriented storage

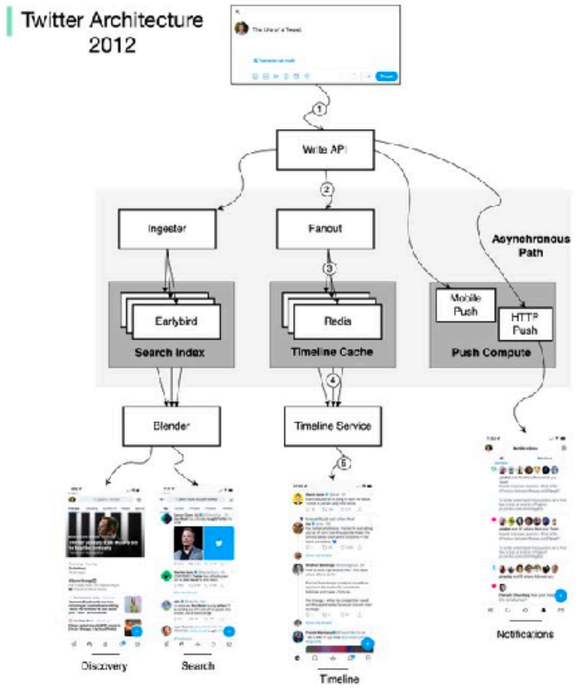
- OLTP v.s. OLAP
- Data warehousing
- Schemas for Analytics
- Column-oriented storage
- Data cubes and materialized views

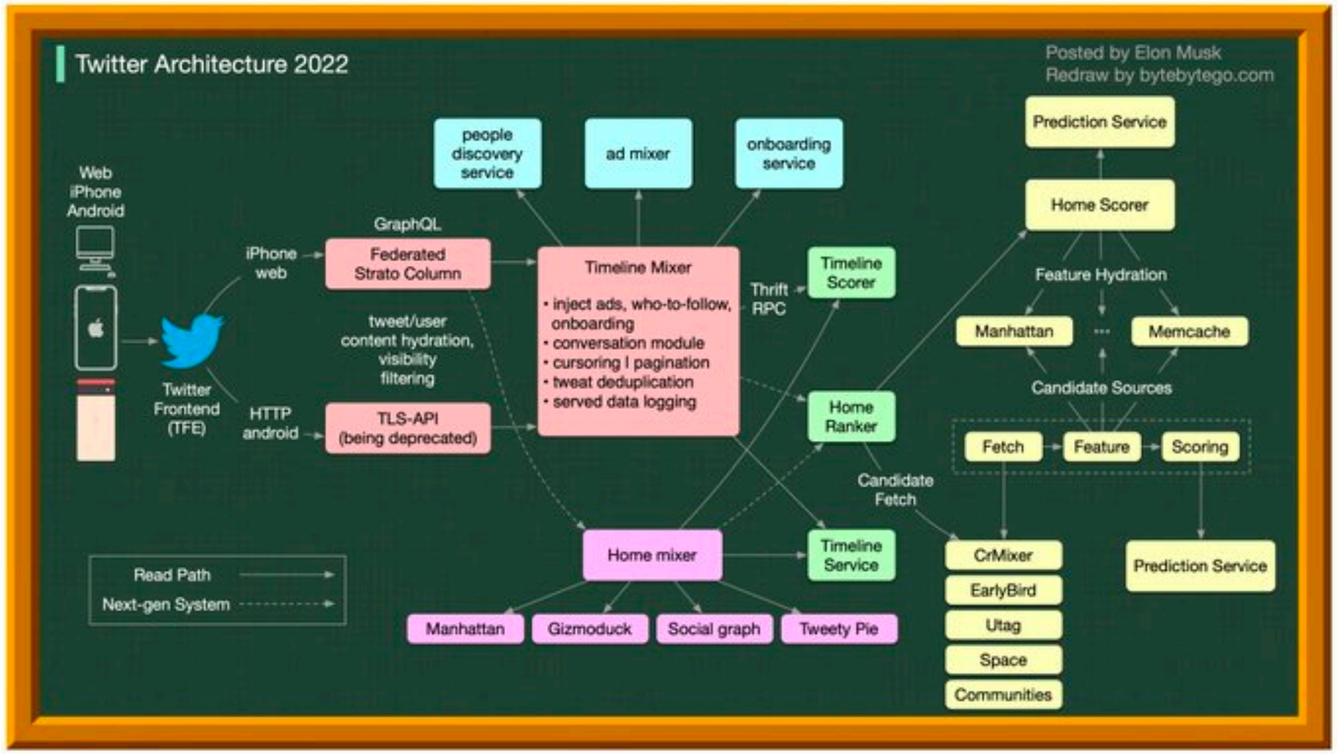
Transaction systems are complex.



Elon Musk's Twitter System Design Diagram Explained
https://www.youtube.com/watch?v=_Y5aGCOkymQ

Transaction systems need to be highly available.



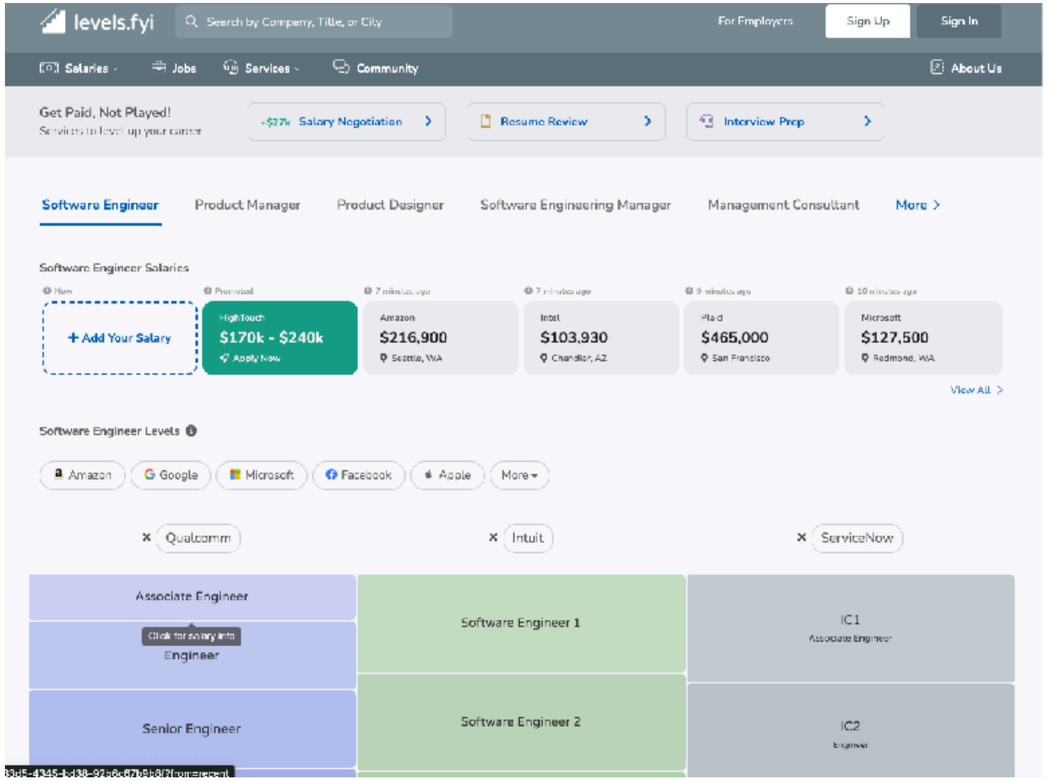


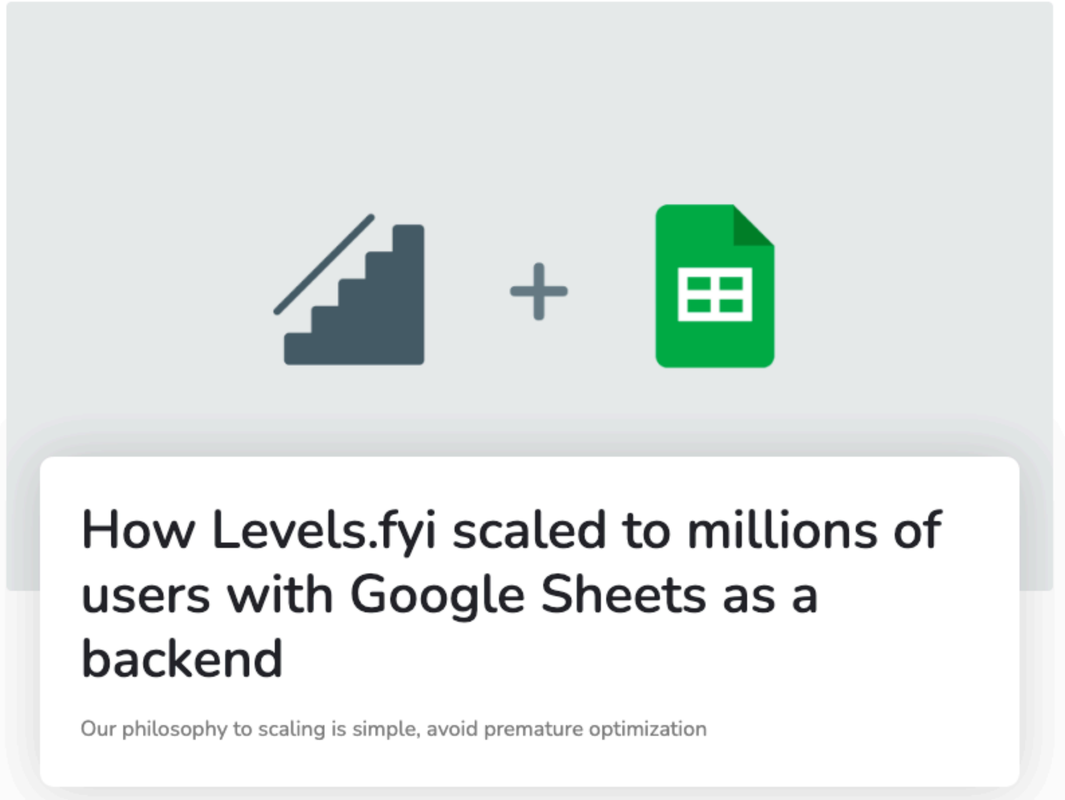
- Low latency.
- Highly available.
- Ad hoc analytic queries are expensive. https://twitter.com/alexxubyte/status/1594008281340530688

Data warehouse

- A separate database that analysts can query to their hearts' content, without affecting OLTP operations.
- Maintain a read-only copy for analytic purposes.
- Only exist in almost all large enterprises.

Small companies?

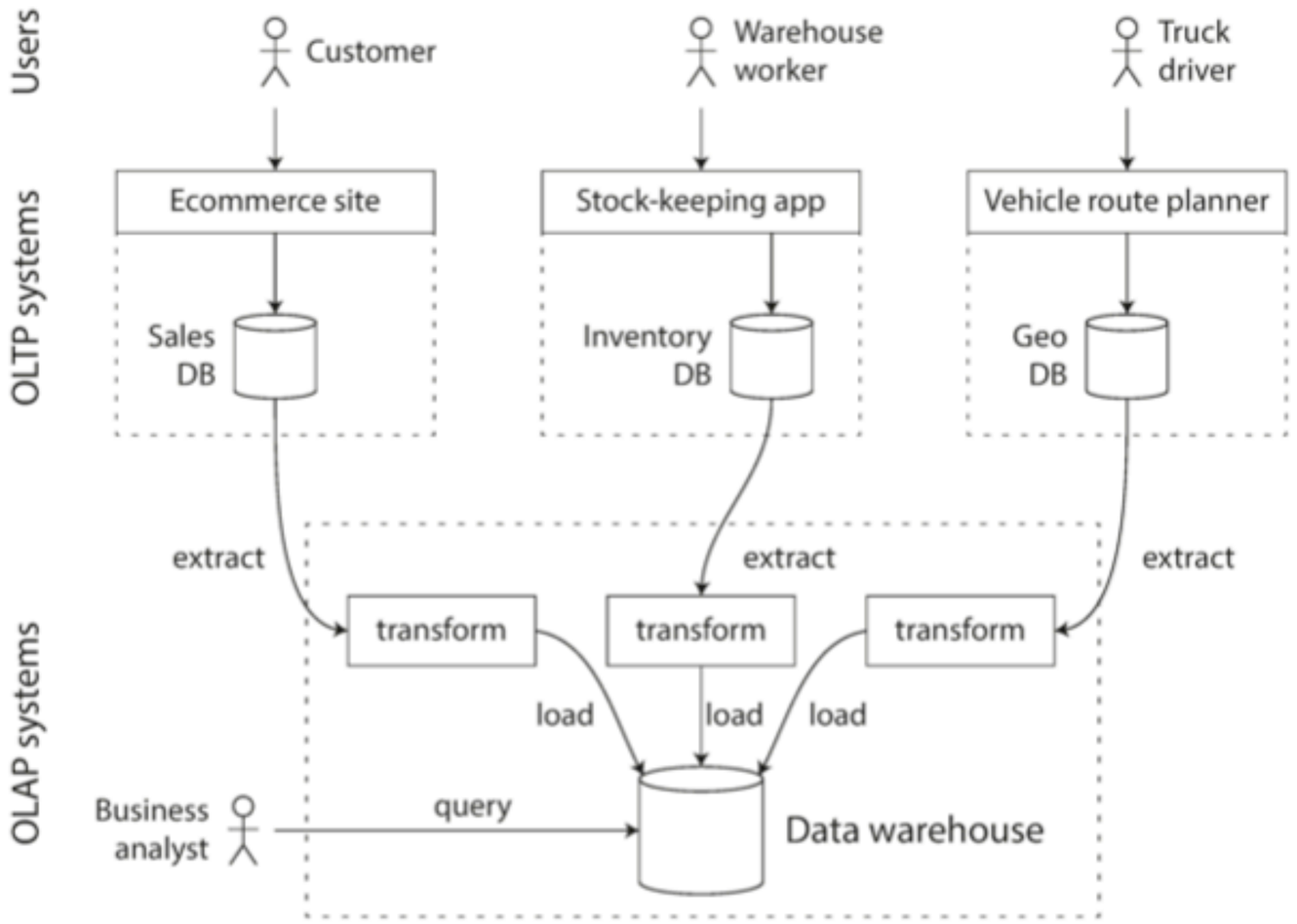




https://www.levels.fyi/blog/scaling-to-millions-with-google-sheets.html

Extract-Transform-Load (ETL)

- Extract
  - Periodica data dump
  - Continuous streaming
- Transform
  - Analysis-friendly schema
  - Data cleaning
- Load into a data warehouse



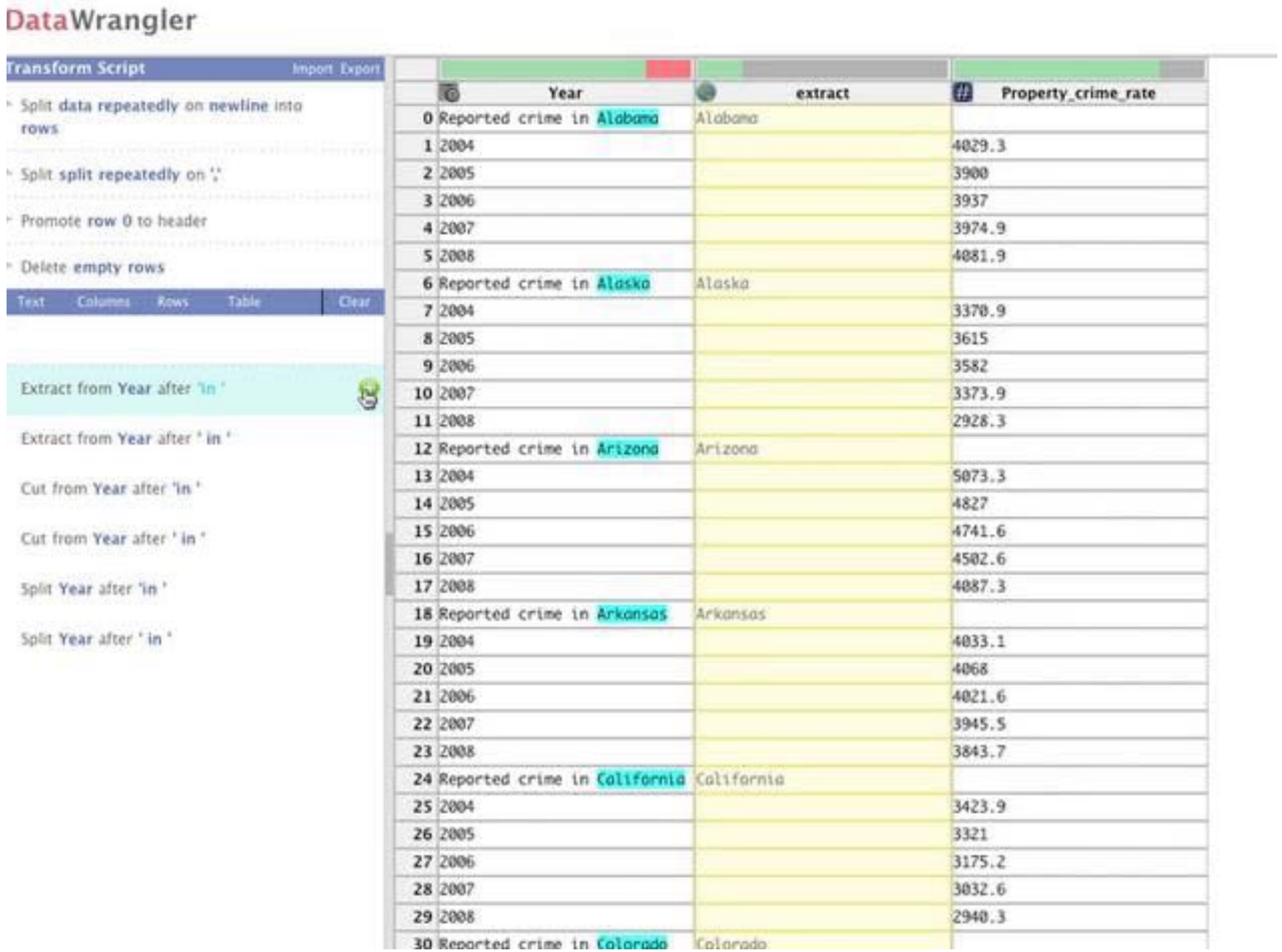
Why data warehouse?

- Separation of concerns
  - Performance (reliability, latency)
  - Expertise requirement, management
- The indexes in last lecture (e.g., SSTable, B-tree) are not good for reading and writing a single record.
  - But are not good at answering analytic queries.

How do you interact with OLAP & OLTP

- SQL query interface
  - *Select * from*
  - "A database system can be considered mature when it has an SQL query interface".
  - Both OLAP and OLTP
- OLAP:
  - More and more codeless user interfaces.

Data wrangler



	Year	extract	Property_crime_rate
0	Reported crime in Alabama	Alabama	
1	2004		4029.3
2	2005		3900
3	2006		3937
4	2007		3974.9
5	2008		4081.9
6	Reported crime in Alaska	Alaska	
7	2004		3370.9
8	2005		3615
9	2006		3582
10	2007		3373.9
11	2008		2928.3
12	Reported crime in Arizona	Arizona	
13	2004		5073.3
14	2005		4827
15	2006		4741.6
16	2007		4502.6
17	2008		4087.3
18	Reported crime in Arkansas	Arkansas	
19	2004		4033.1
20	2005		4068
21	2006		4021.6
22	2007		3945.5
23	2008		3843.7
24	Reported crime in California	California	
25	2004		3423.9
26	2005		3321
27	2006		3175.2
28	2007		3032.6
29	2008		2940.3
30	Reported crime in Colorado	Colorado	

SIGCHI 2011

https://www.trifacta.com/

$400 million, Feb 2022

Today's topic: Column-oriented storage

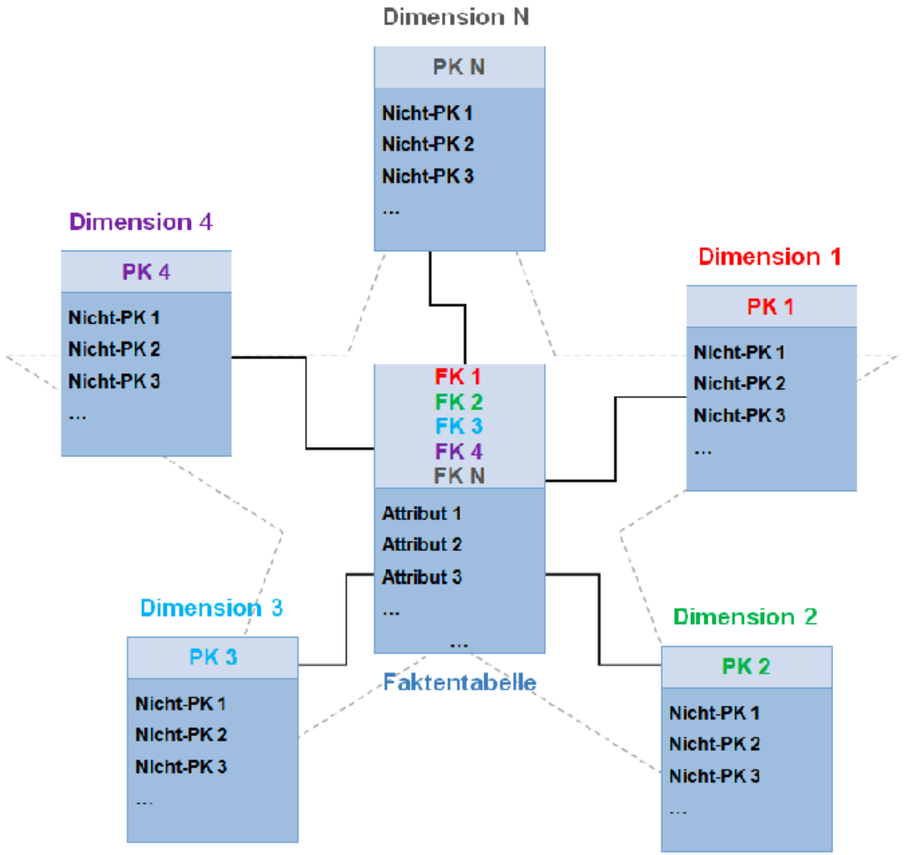
- OLTP v.s. OLAP
- Data warehousing
- Schemas for Analytics
- Column-oriented storage
- Data cubes and materialized views

Data analytic queries

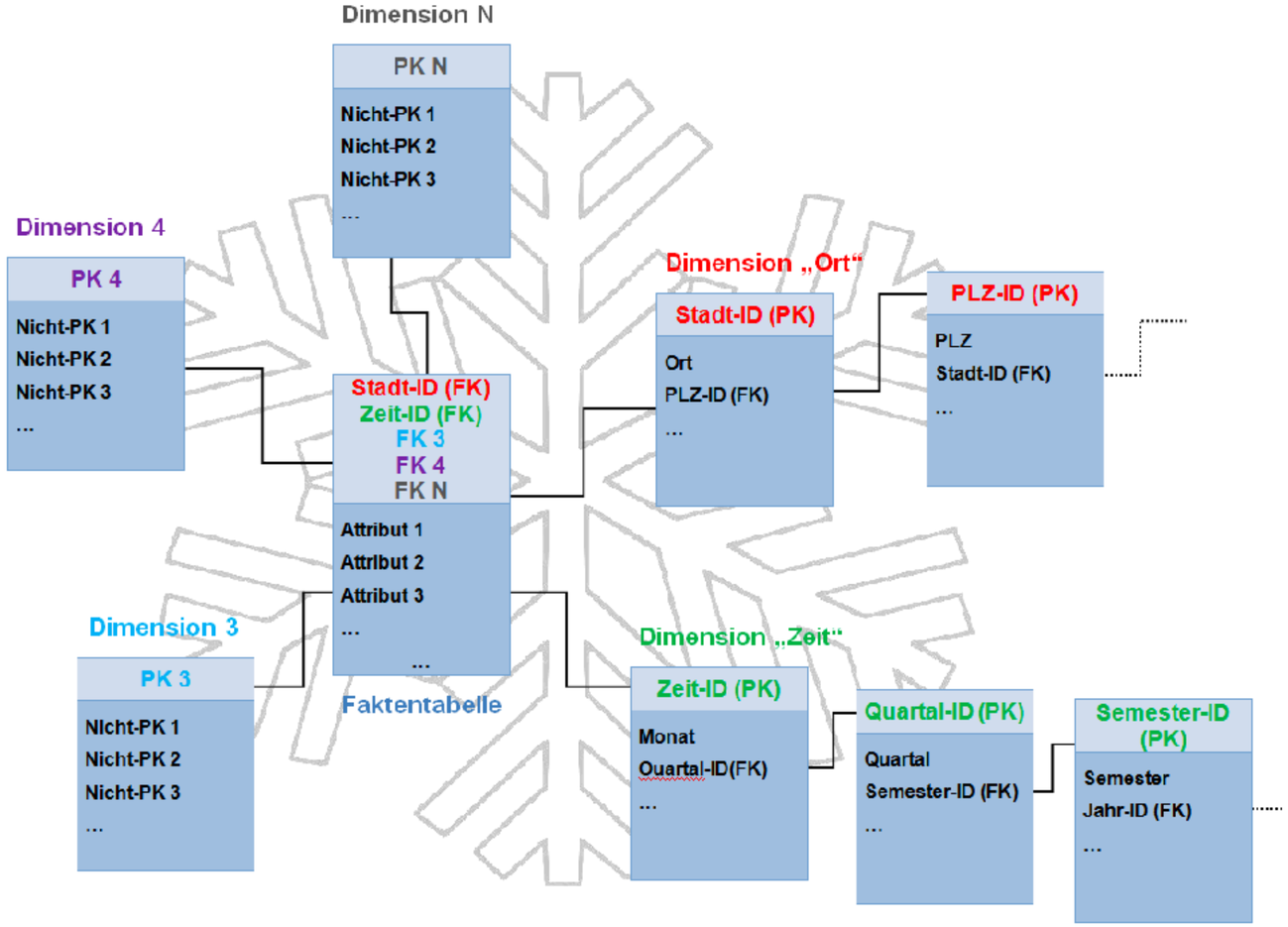
- What was the total revenue of each of our stores in Jan?
- How many more bananas that usual did we sell during our latest data?
- Which brand of baby food is most often purchased together with brand X diapers?

Data model

- Relation (SQL)
- Document (NoSQL)
- Graph (GraphQL)
- Network
- Hierarchy
- Stars
- Snowflake



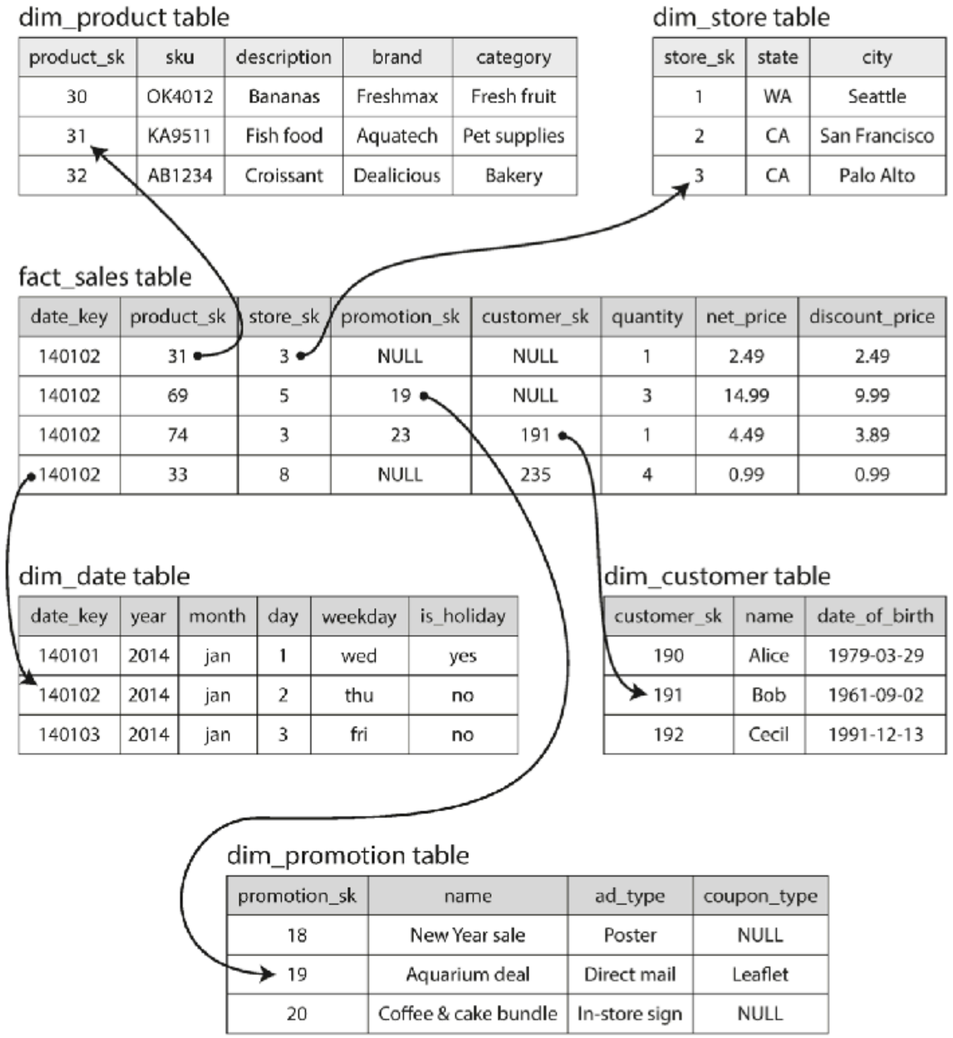
Star



Snowflake

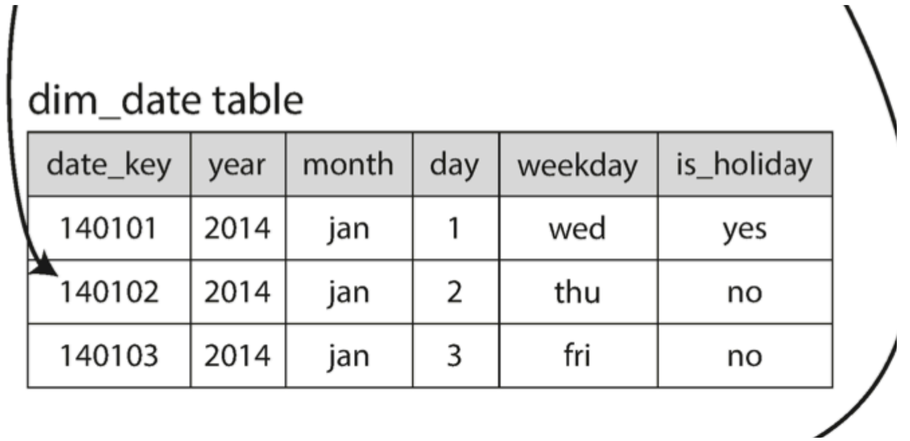
Star schema

- Fact table in the middle
  - A collection of events
  - e.g., click events, page views, retail sales
  - Two types of columns
    - Attributes
    - References to dimension tables.
- Fact table: event meta data
- Dimensions: who, what, where, when, how, and why of the event.



Example: dim_date table

- Speed up the analysis.
- Easier development.

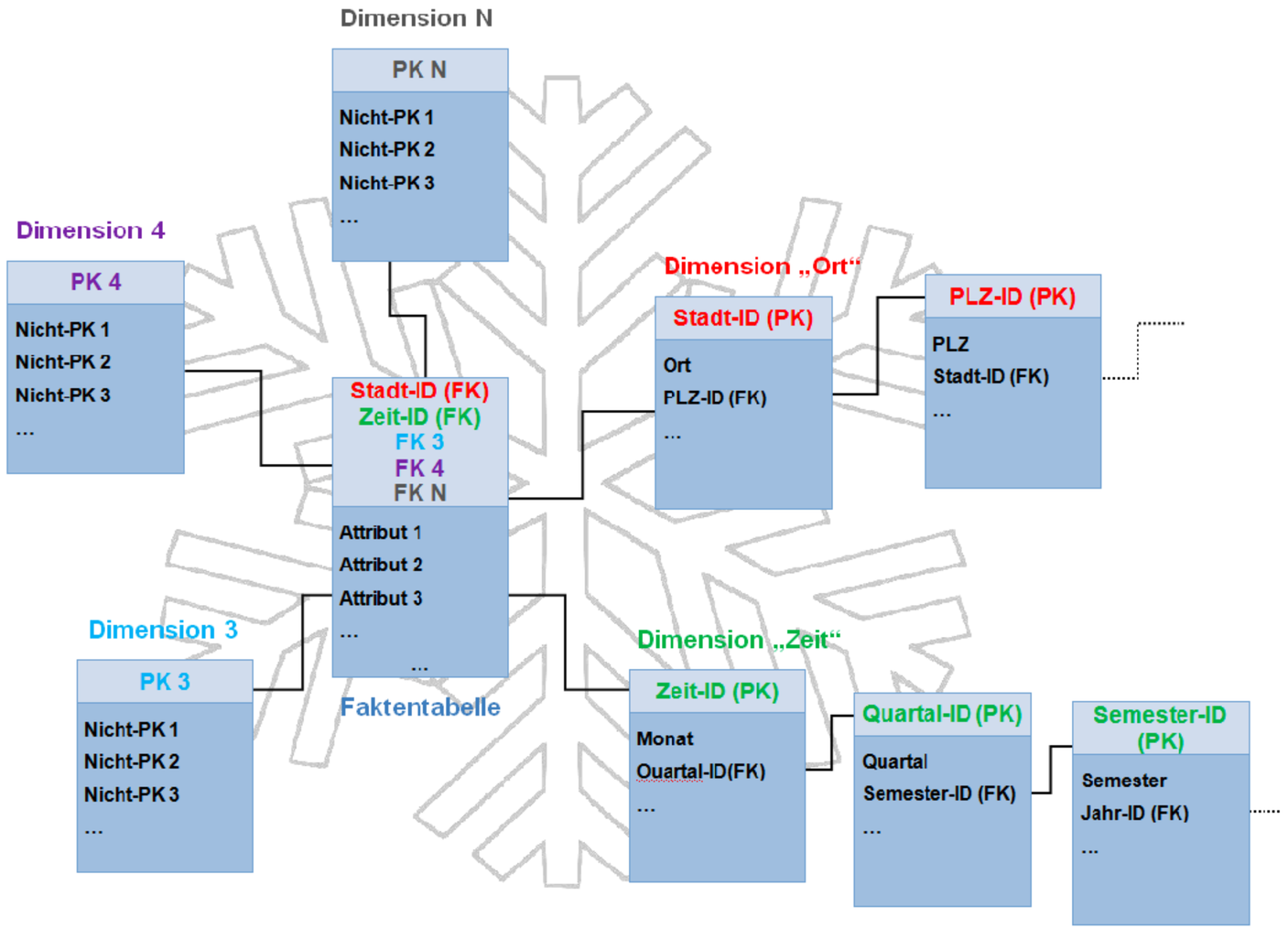


dim_date table

date_key	year	month	day	weekday	is_holiday
140101	2014	jan	1	wed	yes
140102	2014	jan	2	thu	no
140103	2014	jan	3	fri	no

Snowflake schema

- More normalized data model



Database normalization

- Storage redundancy
- Easy interactions.

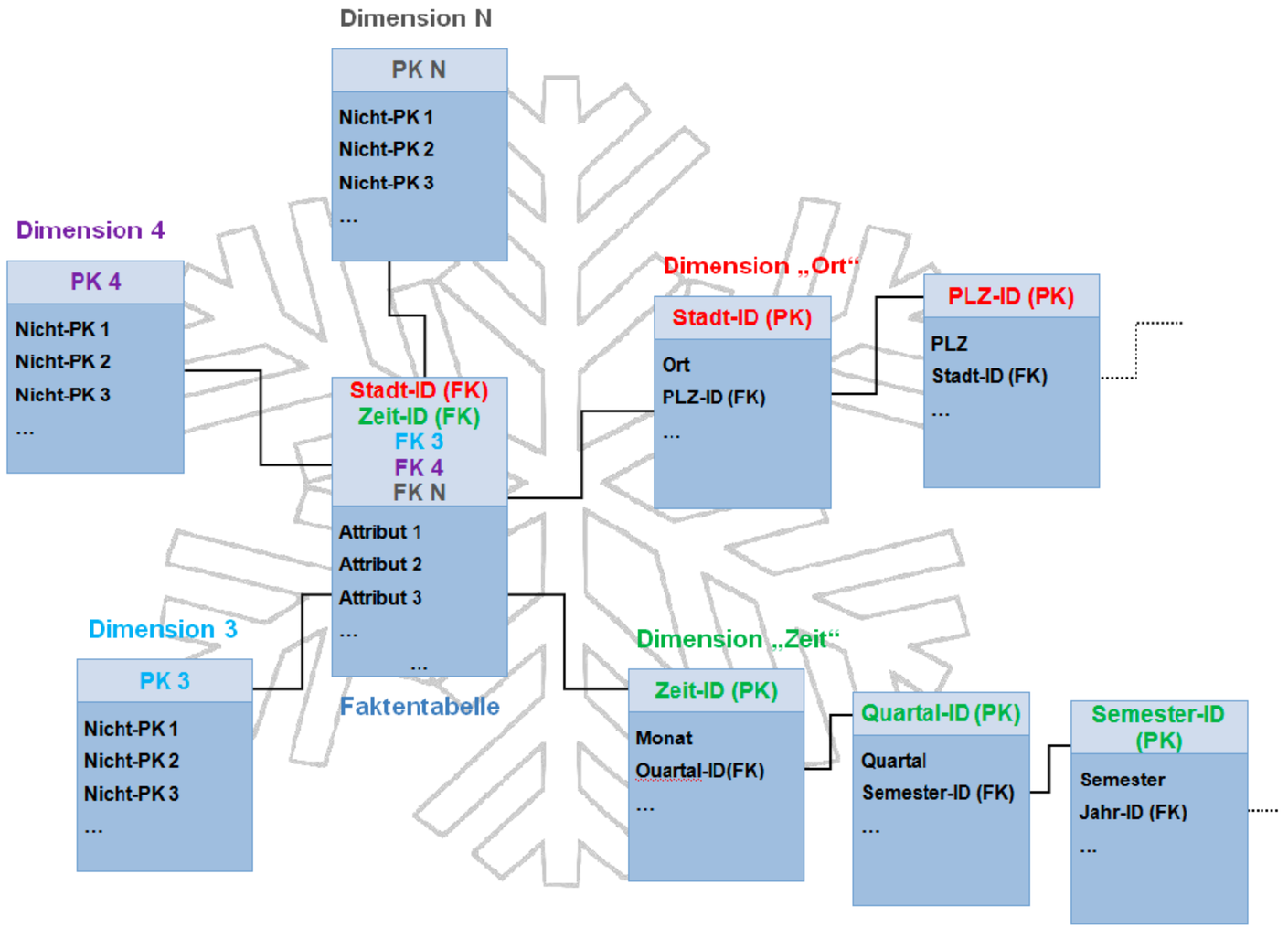
EmpID	Employee	Age	Dept
1001	ABC	30	Sales,Finance
1002	CDE	30	Sales,Finance,DevOps

DeptID	DeptName
1	Sales
2	Finance
3	DevOps

EmpId	Employee	Age	DepID
1001	ABC	30	1
1001	ABC	30	2
1002	CDE	40	1
1002	CDE	40	2
1002	CDE	40	3

Snowflake schema tradeoffs

- More storage efficiency
- Enforces data quality
- Slower
- Lots of overhead upon initial setup
- High maintenance costs



Today's topic: Column-oriented storage

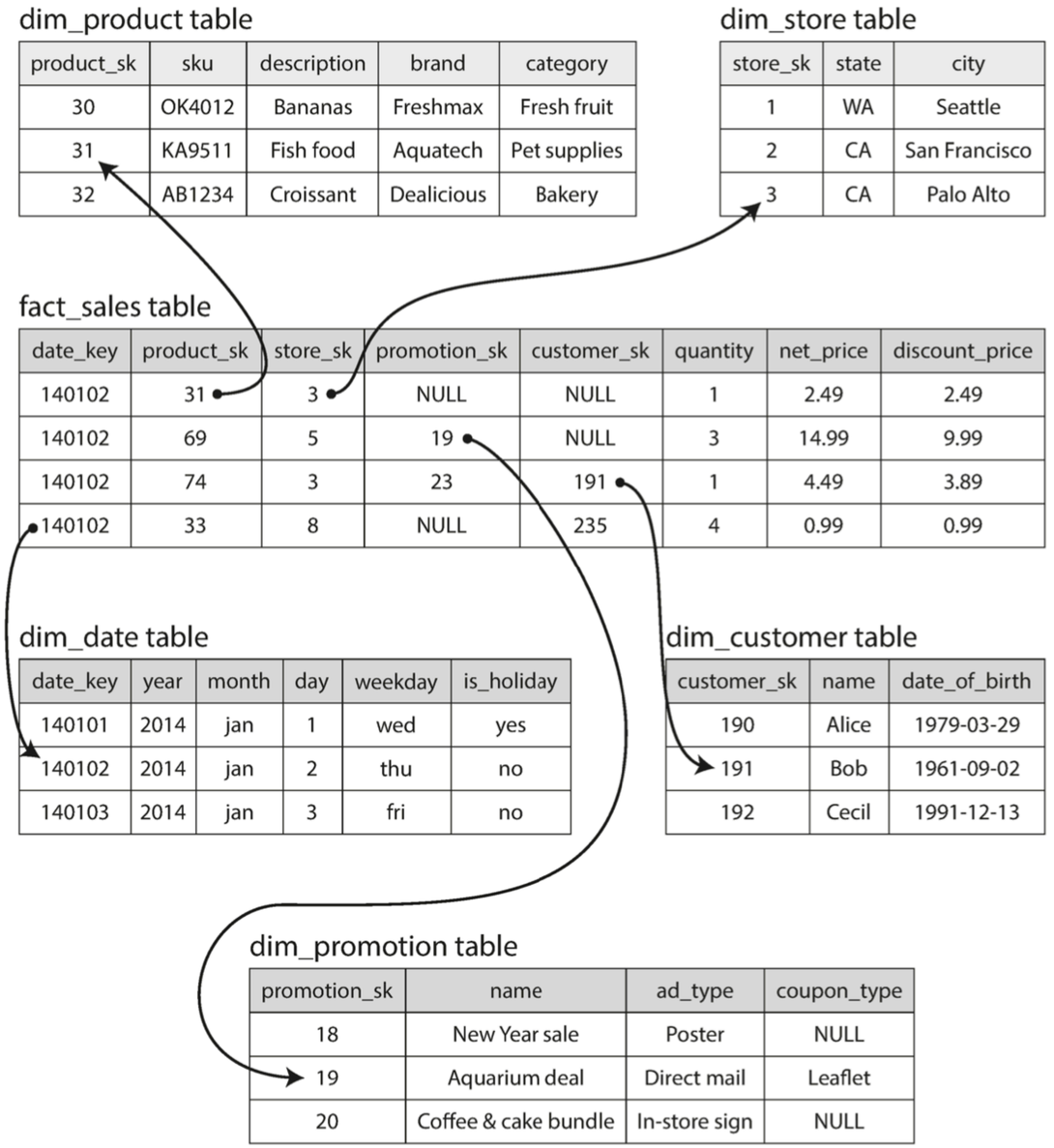
- OLTP v.s. OLAP
- Data warehousing
- Schemas for Analytics
- Column-oriented storage
- Data cubes and materialized views

Data scale

- Fact tables
  - Hundreds of columns
  - Trillions of rows
  - Petabytes of data
- Dimension tables
  - Million of rows.
  - Can be wide. But less common.

How many columns do we need?

- What was the total revenue of each of our stores in Jan?
- How many more bananas that usual did we sell during our latest data?
- Which brand of baby food is most often purchased together with brand X diapers?



dim_product table

product_sk	sku	description	brand	category
30	OK4012	Bananas	Freshmax	Fresh fruit
31	KA9511	Fish food	Aquatech	Pet supplies
32	AB1234	Croissant	Dealicious	Bakery

dim_store table

store_sk	state	city
1	WA	Seattle
2	CA	San Francisco
3	CA	Palo Alto

fact_sales table

date_key	product_sk	store_sk	promotion_sk	customer_sk	quantity	net_price	discount_price
140102	31	3	NULL	NULL	1	2.49	2.49
140102	69	5	19	NULL	3	14.99	9.99
140102	74	3	23	191	1	4.49	3.89
140102	33	8	NULL	235	4	0.99	0.99

dim_date table

date_key	year	month	day	weekday	is_holiday
140101	2014	jan	1	wed	yes
140102	2014	jan	2	thu	no
140103	2014	jan	3	fri	no

dim_customer table

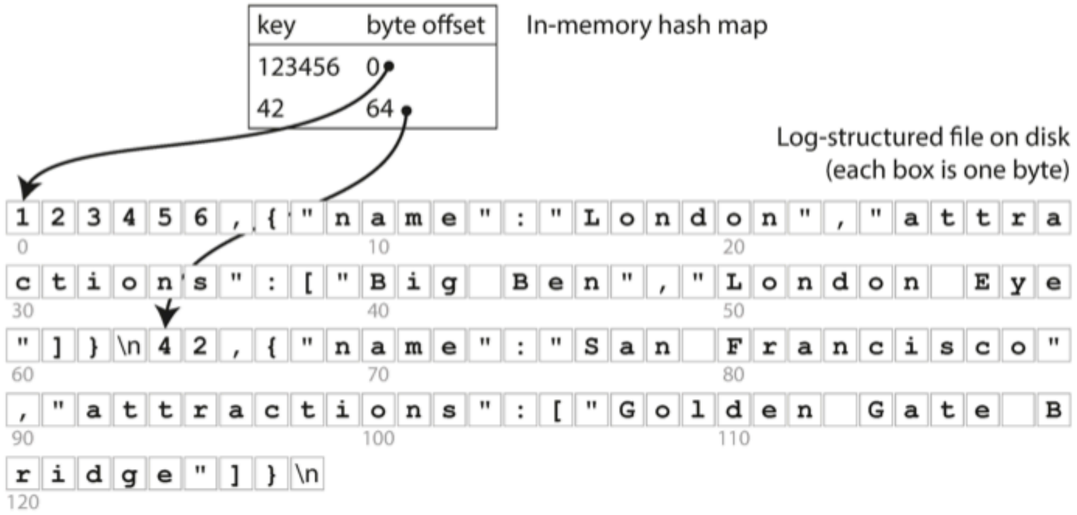
customer_sk	name	date_of_birth
190	Alice	1979-03-29
191	Bob	1961-09-02
192	Cecil	1991-12-13

dim_promotion table

promotion_sk	name	ad_type	coupon_type
18	New Year sale	Poster	NULL
19	Aquarium deal	Direct mail	Leaflet
20	Coffee & cake bundle	In-store sign	NULL

How does a Row-oriented storage work?

- Storage:
  - All the values from one row of a table are stored next to each other.
- What was the total revenue in January?
  - Load indexes into the memory
  - Find all the records in January.
  - Load all of these rows (100+ attributes) from disk into memory
  - Parse
  - Filter



Implementations for column-oriented

Row-oriented

ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented

Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003

fact_sales table

date_key	product_sk	store_sk	promotion_sk	customer_sk	quantity	net_price	discount_price
140102	69	4	NULL	NULL	1	13.99	13.99
140102	69	5	19	NULL	3	14.99	9.99
140102	69	5	NULL	191	1	14.99	14.99
140102	74	3	23	202	5	0.99	0.89
140103	31	2	NULL	NULL	1	2.49	2.49
140103	31	3	NULL	NULL	3	14.99	9.99
140103	31	3	21	123	1	49.99	39.99
140103	31	8	NULL	233	1	0.99	0.99

Columnar storage layout:

date_key file contents:	140102, 140102, 140102, 140102, 140103, 140103, 140103, 140103
product_sk file contents:	69, 69, 69, 74, 31, 31, 31, 31
store_sk file contents:	4, 5, 5, 3, 2, 3, 3, 8
promotion_sk file contents:	NULL, 19, NULL, 23, NULL, NULL, 21, NULL
customer_sk file contents:	NULL, NULL, 191, 202, NULL, NULL, 123, 233
quantity file contents:	1, 3, 1, 5, 1, 3, 1, 1
net_price file contents:	13.99, 14.99, 14.99, 0.99, 2.49, 14.99, 49.99, 0.99
discount_price file contents:	13.99, 9.99, 14.99, 0.89, 2.49, 9.99, 39.99, 0.99

How does a column-oriented storage work?

- Storage:
  - Store all the values from each column together instead.
- What was the total revenue in January?
  - Load indexes into the memory
  - Find all the records in January.
  - Load all of these rows (~~100+ attributes~~ -> 1 row) from disk into memory
    - 100 times improvement
  - Parse
  - Filter

fact_sales table

date_key	product_sk	store_sk	promotion_sk	customer_sk	quantity	net_price	discount_price
140102	69	4	NULL	NULL	1	13.99	13.99
140102	69	5	19	NULL	3	14.99	9.99
140102	69	5	NULL	191	1	14.99	14.99
140102	74	3	23	202	5	0.99	0.89
140103	31	2	NULL	NULL	1	2.49	2.49
140103	31	3	NULL	NULL	3	14.99	9.99
140103	31	3	21	123	1	49.99	39.99
140103	31	8	NULL	233	1	0.99	0.99

Columnar storage layout:

date_key file contents:	140102, 140102, 140102, 140102, 140103, 140103, 140103, 140103
product_sk file contents:	69, 69, 69, 74, 31, 31, 31, 31
store_sk file contents:	4, 5, 5, 3, 2, 3, 3, 8
promotion_sk file contents:	NULL, 19, NULL, 23, NULL, NULL, 21, NULL
customer_sk file contents:	NULL, NULL, 191, 202, NULL, NULL, 123, 233
quantity file contents:	1, 3, 1, 5, 1, 3, 1, 1
net_price file contents:	13.99, 14.99, 14.99, 0.99, 2.49, 14.99, 49.99, 0.99
discount_price file contents:	13.99, 9.99, 14.99, 0.89, 2.49, 9.99, 39.99, 0.99

Column compression

- Data in the same column are more repetitive.
  - SSTable?
  - Index v.s. Values
- Save storage space
- Improve I/O bandwidth usage

fact_sales table

date_key	product_sk	store_sk	promotion_sk	customer_sk	quantity	net_price	discount_price
140102	69	4	NULL	NULL	1	13.99	13.99
140102	69	5	19	NULL	3	14.99	9.99
140102	69	5	NULL	191	1	14.99	14.99
140102	74	3	23	202	5	0.99	0.89
140103	31	2	NULL	NULL	1	2.49	2.49
140103	31	3	NULL	NULL	3	14.99	9.99
140103	31	3	21	123	1	49.99	39.99
140103	31	8	NULL	233	1	0.99	0.99

Columnar storage layout:

date_key file contents:	140102, 140102, 140102, 140102, 140103, 140103, 140103, 140103
product_sk file contents:	69, 69, 69, 74, 31, 31, 31, 31
store_sk file contents:	4, 5, 5, 3, 2, 3, 3, 8
promotion_sk file contents:	NULL, 19, NULL, 23, NULL, NULL, 21, NULL
customer_sk file contents:	NULL, NULL, 191, 202, NULL, NULL, 123, 233
quantity file contents:	1, 3, 1, 5, 1, 3, 1, 1
net_price file contents:	13.99, 14.99, 14.99, 0.99, 2.49, 14.99, 49.99, 0.99
discount_price file contents:	13.99, 9.99, 14.99, 0.89, 2.49, 9.99, 39.99, 0.99

Bitmap encoding

Column values:

product_sk:	69	69	69	69	74	31	31	31	31	29	30	30	31	31	31	68	69	69

Bitmap for each possible value:

product_sk = 29:	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
product_sk = 30:	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
product_sk = 31:	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0
product_sk = 68:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
product_sk = 69:	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
product_sk = 74:	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

- Many transactions
- A small amount of distinct values
- Transform them into bitmaps
  - Bitmap => One unique value
  - Bit => Occurrence & Order

Bitmap encoding

Column values:

product_sk:	69	69	69	69	74	31	31	31	31	29	30	30	31	31	31	68	69	69

Bitmap for each possible value:

product_sk = 29:	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
product_sk = 30:	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
product_sk = 31:	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0
product_sk = 68:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
product_sk = 69:	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
product_sk = 74:	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Run-length encoding:

product_sk = 29:	9, 1	(9 zeros, 1 one, rest zeros)
product_sk = 30:	10, 2	(10 zeros, 2 ones, rest zeros)
product_sk = 31:	5, 4, 3, 3	(5 zeros, 4 ones, 3 zeros, 3 ones, rest zeros)
product_sk = 68:	15, 1	(15 zeros, 1 one, rest zeros)
product_sk = 69:	0, 4, 12, 2	(0 zeros, 4 ones, 12 zeros, 2 ones)
product_sk = 74:	4, 1	(4 zeros, 1 one, rest zeros)

- Many transactions
- A small amount of distinct values
- Transform them into bitmaps
  - Bitmap => One unique value
  - Bit => Occurrence & Order
- Run-length encoding for sparse bitmaps

Bitmap indexes for queries

`WHERE product_sk IN (30, 68, 69):`

Load the three bitmaps for `product_sk = 30`, `product_sk = 68`, and `product_sk = 69`, and calculate the bitwise *OR* of the three bitmaps, which can be done very efficiently.

`WHERE product_sk = 31 AND store_sk = 3:`

Load the bitmaps for `product_sk = 31` and `store_sk = 3`, and calculate the bitwise *AND*. This works because the columns contain the rows in the same order, so the *k*th bit in one column's bitmap corresponds to the same row as the *k*th bit in another column's bitmap.

Bitwise operations?

Bottlenecks in the I/O performance

- Want to backup 10,000 1MB files to an external hard drive?
- Want to download 1,000 1MB files from the Cloud (e.g., AWS S3) to your computer?

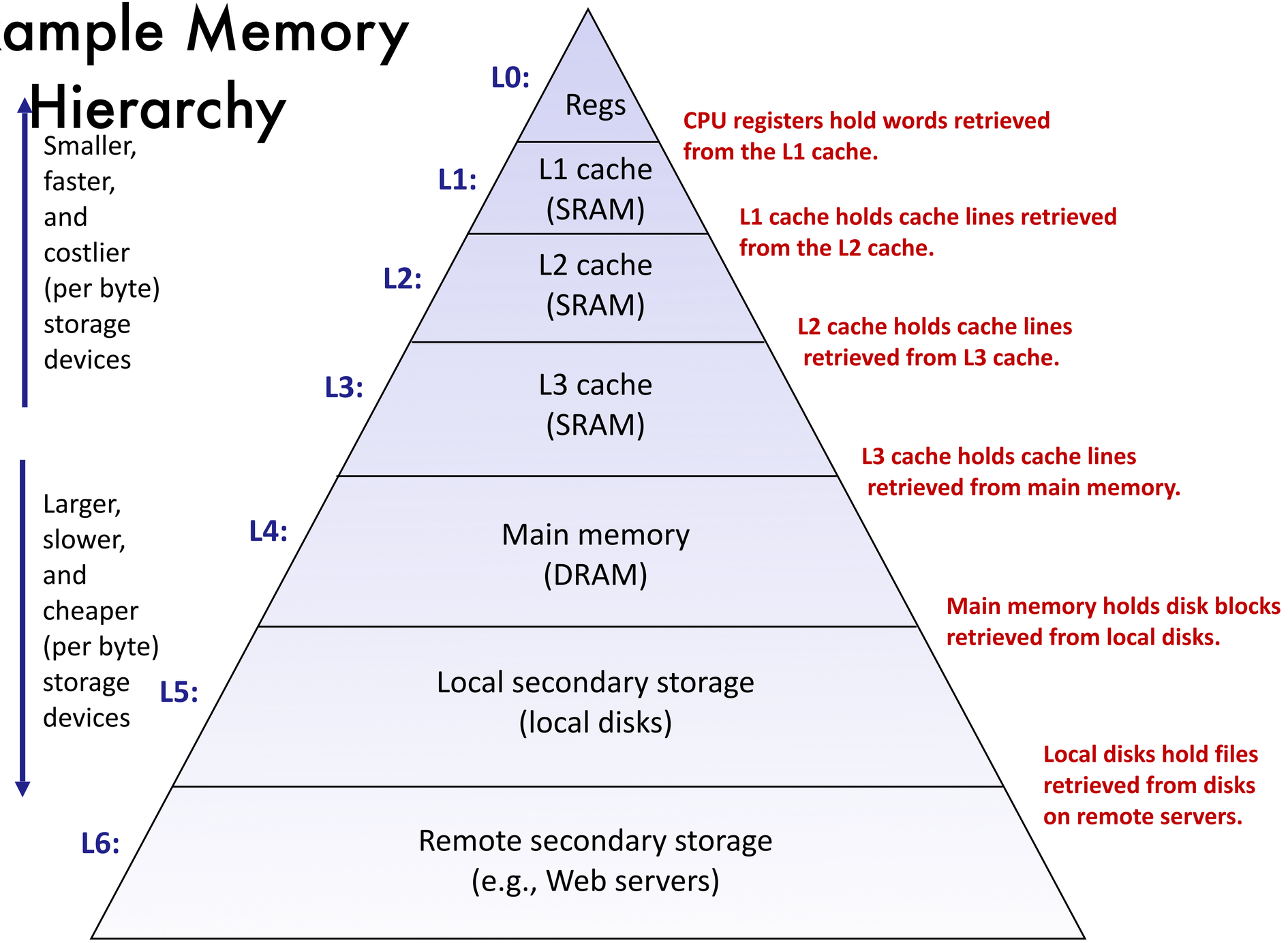
Bottlenecks in the I/O performance

- Want to backup 10,000 1MB files to an external hard drive?
  - Read the first file's information from the disk directory
  - Locate the first file on disk
  - Locate free space on the destination.
  - Write the first file's directory information on the destination
  - Read the first fie into RAM
  - Write what you've read to the destination
  - Close the first file on the destination
  - Rlease the first file's handle on the source
  - Do it for 10,000 times.

Bottlenecks in the I/O performance

- Want to backup 10,000 1MB files to an external hard drive?
  - Compress all files into one mega file first.
    - => The file would be smaller than 10 GB due to compression.
  - Procedure:
    - Read the file's information from the disk directory
    - Locate the file on disk
    - Locate free space on the destination.
    - Write the file's directory information on the destination
    - Read as much of the file as will fit in RAM
      - Repeat it for X times.
    - Write what you've read to the destination
    - Close the file on the destination
    - Release first file's handle on the source

Example Memory Hierarchy



Common bottlenecks

- Memory hierarchy
  - Compression
  - Cache
- Storage patterns
- Encoding and decoding
  - Bitwise operation is preferred.

Other compressions

Log message

2020-01-02T03:04:05.006 INFO Task task_12 assigned to container: [NodeAddr
ess:172.128.0.41, ContainerID:container_15], operation took 0.335 seconds

Variable dictionary

ID	Schema	Ptr
0	task_\d+	
1	\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}	
2	container_\d+	

ID	Variable value	Segments
8	task_12	...

ID	Variable value	Segments
9	172.128.0.41	...

ID	Variable value	Segments
9	container_15	...

Log type dictionary

ID	Log type	Segments
4	INFO Task \x11\x00 assigned to container: [NodeAddress:\x11\x01, ContainerID:\x11\x02], operation took \x12\x13 seconds	...

Encoded messages

Timestamp	Log type ID	Variable values
1577934245006	4	8 9 9 0x3FD570A3D70A3D71

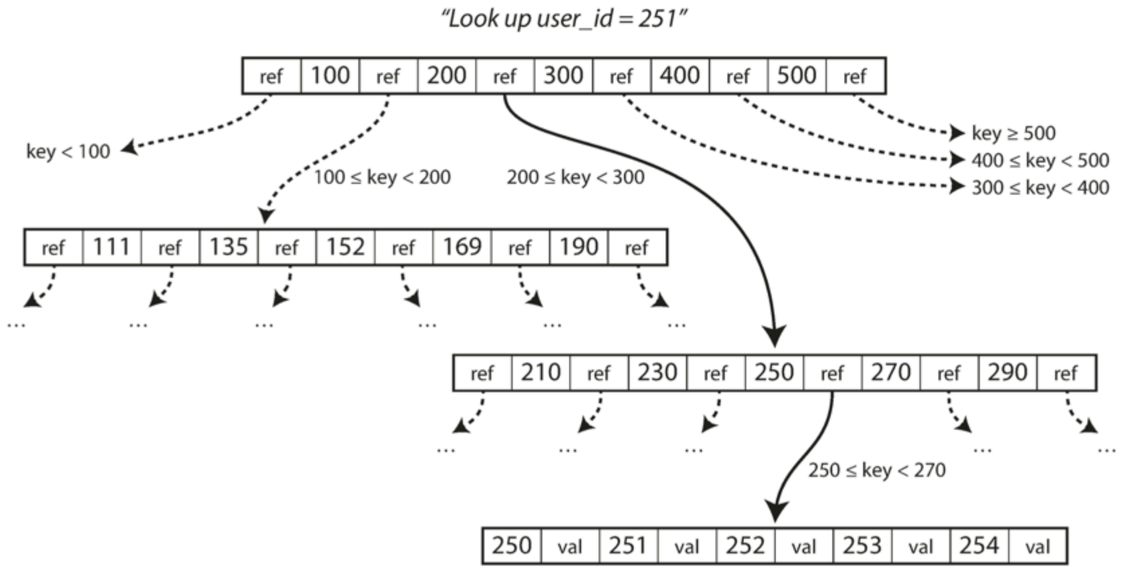
Figure 3: A log message and its encoding. Dictionary variables are in blue; Non-dictionary variables are in orange.

See more in https://www.usenix.org/system/files/osdi21-rodrigues.pdf

Sort order in column storage

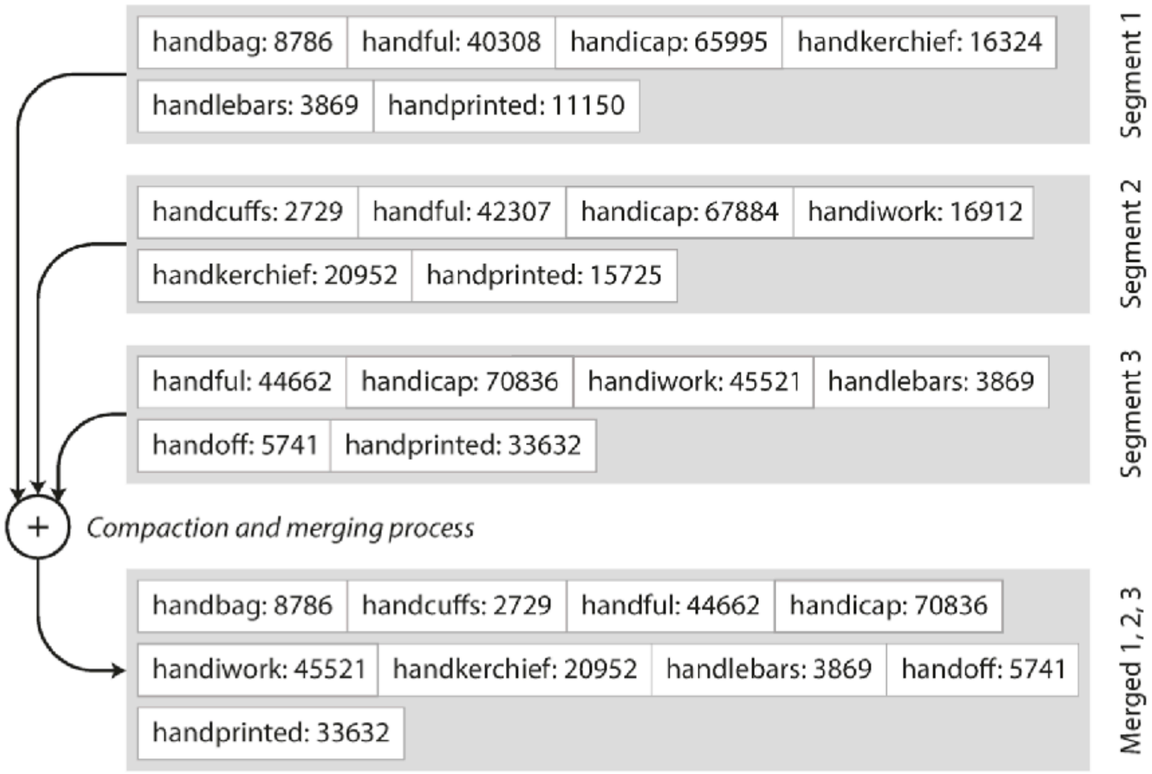
- If queries often target date ranges, such as the last month,
  - Sort the rows by the *date_key.*
  - Note sorting a column independently is meaningless.
- Multiple sort keys
  - e.g., First: *date_key*, Second: *product_ski*, Third, …
- Compression effect
- Challenge?
  - How to write to a sorted column-oriented storage?

Recap: B-tree



- Update-in-place will not work
  - Insert a raw in the middle of a sorted tables

Recap: SSTable (sorted string table)



- LSM-tree for Column storage
  - All writes -> an in-memory storage
  - The memory storage can be row or column-oriented
  - Write merged files to the disk
- Query engines handle everything behind the scene
- Important app for SSTable
- Feasible for OLAP.

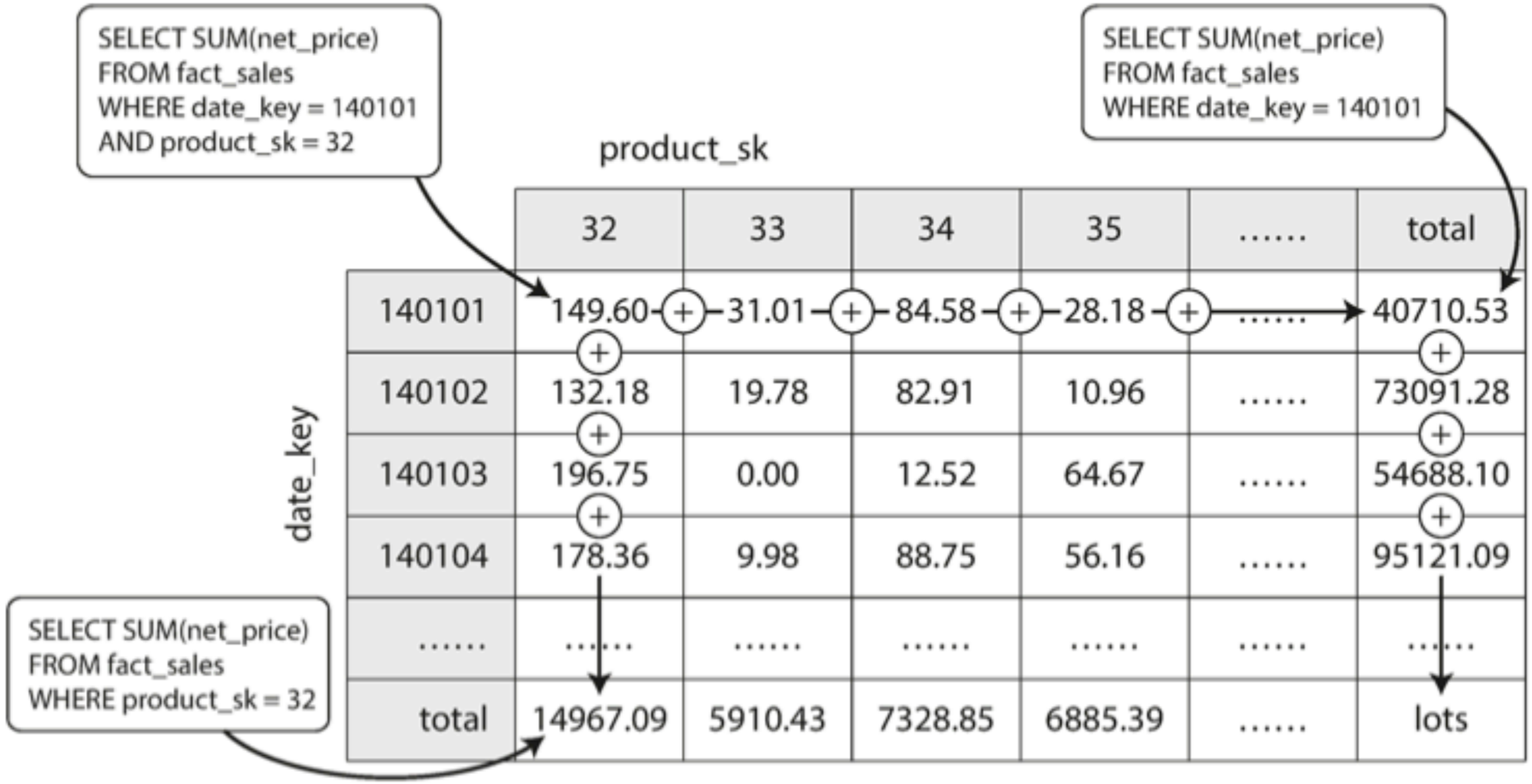
Today's topic: Column-oriented storage

- OLTP v.s. OLAP
- Data warehousing
- Schemas for Analytics
- Column-oriented storage
- Data cubes and materialized views

Materialized aggregates

- A few functions are very common.
  - COUNT, SUM, AVG, MIN, or MAX in SQL
- Why not cache some of the counts or sums that queries use most often?
  - Materialized view:
    - A table-like object whose contents are the results of some query.
    - The object of data would be save to disk for future access.
    - A similar concept: virtual view => expanded queries.

Data cube | OLAP cube | ... | Hypercube



Common bottlenecks

- Memory hierarchy
  - Compression
  - Cache
- Storage patterns
- Encoding and decoding
  - Bitwise operation is preferred.
- **Reuse computations**

Recall: Your role!

- **Well-chosen indexes speed up read queries, but every index slows down writes.**
  - DB do not index everything by default.
  - App developers or db admins choose indexes manually.
    - Based on domain knowledge.
  - Balance the tradeoffs.

Reviews

- DDIA Chapter 3 Storage and Retrieval
- Important concepts: Hashtable indexes, SSTable, LSM, B-tree, In-memory database, OLAP, OLTP, Data warehouse, Schemas for Analytics, Column-oriented storage, compression, sorting.
- What’re the differences between the compression in SSTable and Column-oriented storage?
- Why is B-tree faster than SSTable in reading?
- Why is SSTable faster than B-tree in writing?
- What are the tradeoffs between Stars and Snowflakes?