CHAPTER 3

DESIGNING DATA-INTENSIVE APPLICATIONS

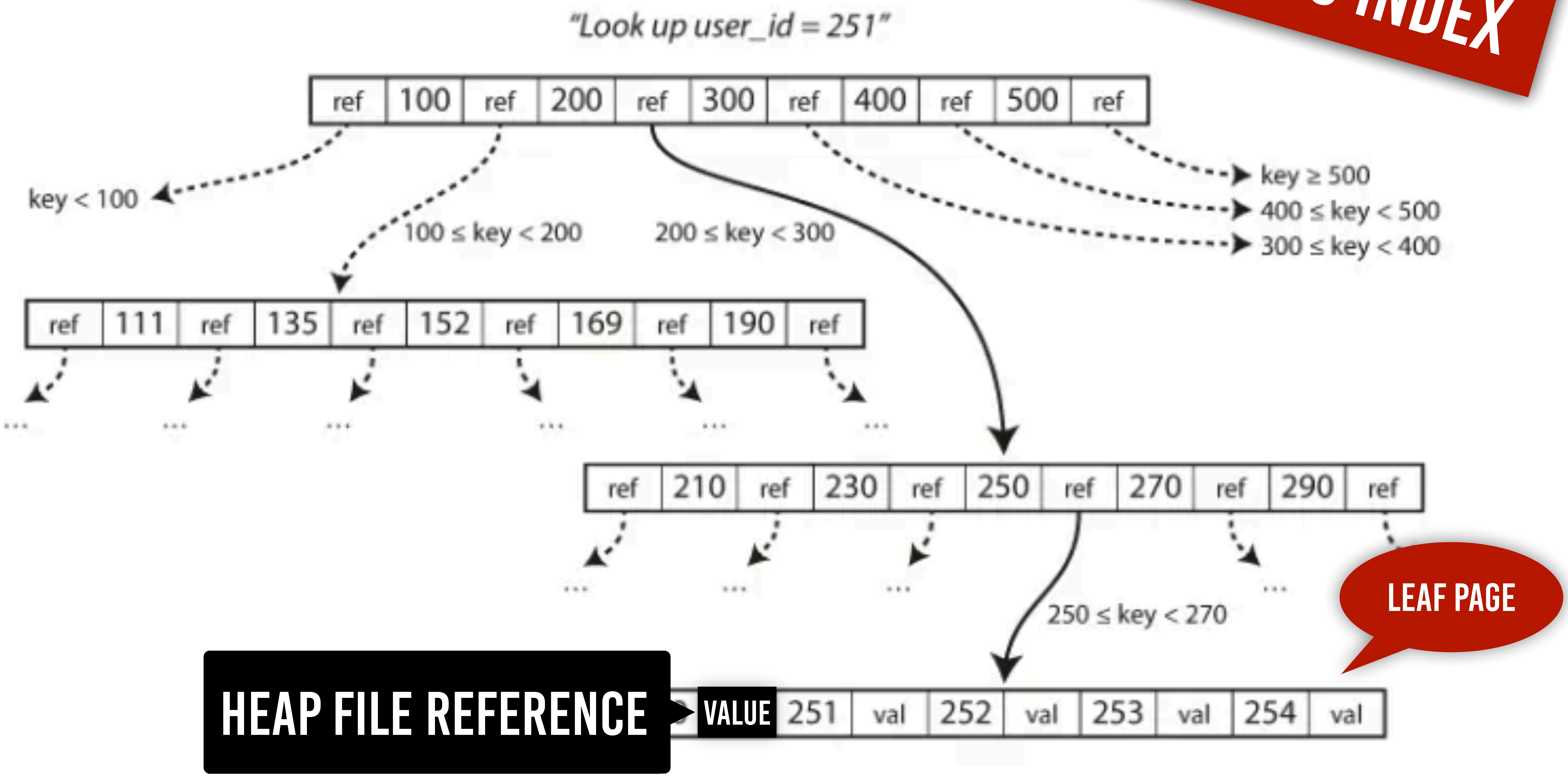
DATABASE INDEX

HOW SECONDARY INDEX WORKS?

COVERING INDEX

B-TREES

TABLE
ID
NAME
USER_ID



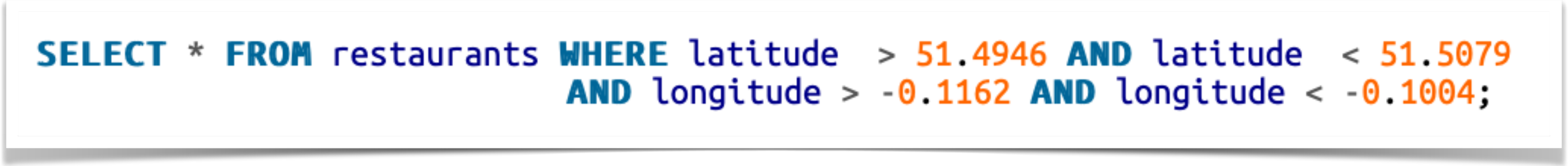
"Look up user_id = 251"
ref 100 ref 200 ref 300 ref 400 ref 500 ref
key < 100
100 ≤ key < 200
200 ≤ key < 300
key ≥ 500
400 ≤ key < 500
300 ≤ key < 400
ref 111 ref 135 ref 152 ref 169 ref 190 ref
...
ref 210 ref 230 ref 250 ref 270 ref 290 ref
250 ≤ key < 270
LEAF PAGE
HEAP FILE REFERENCE
VALUE 251 val 252 val 253 val 254 val

MULTI-COLUMN INDEX

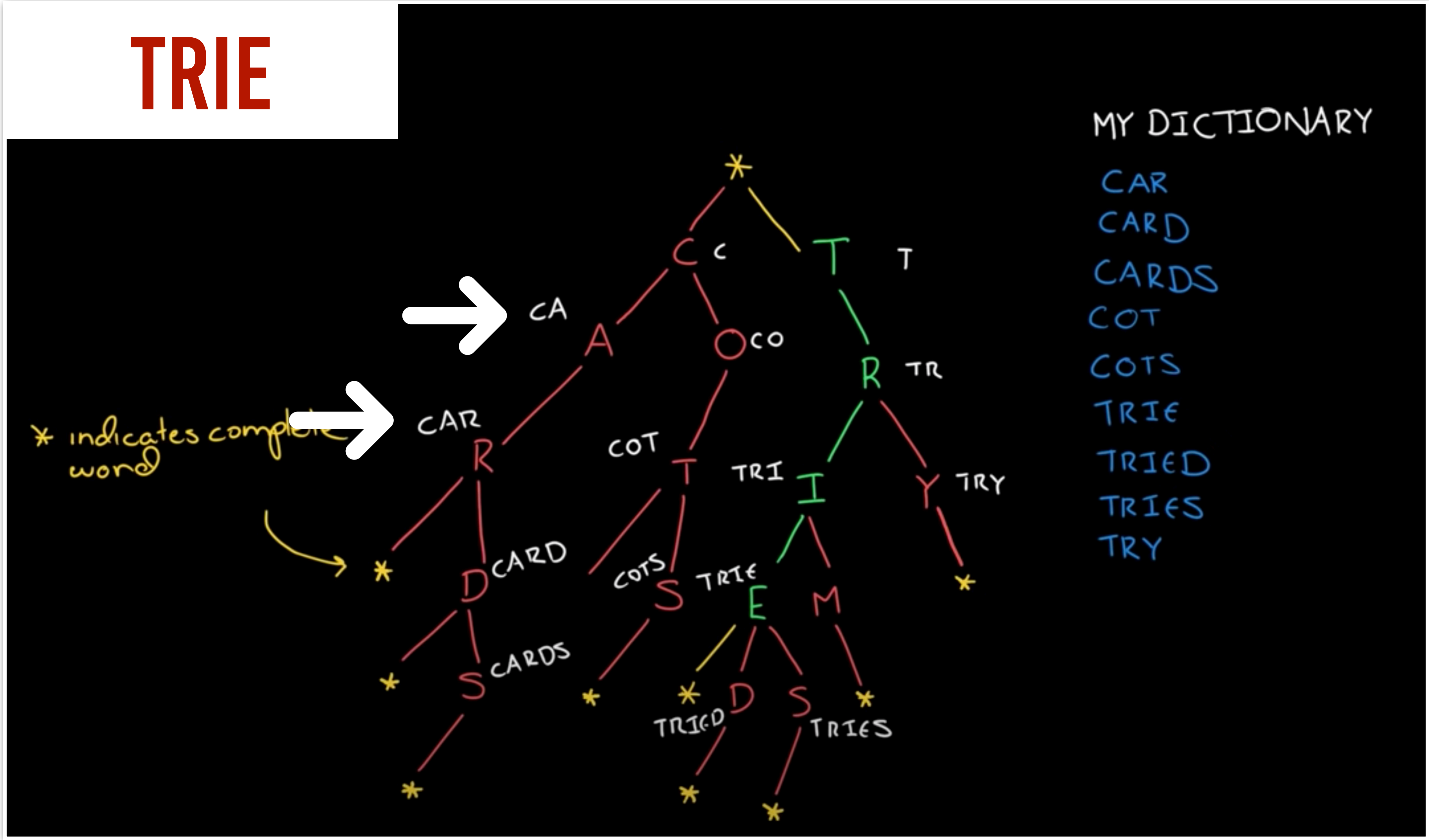
FUZZY INDEX

TABLE
ID
FIRST_NAME
LAST_NAME

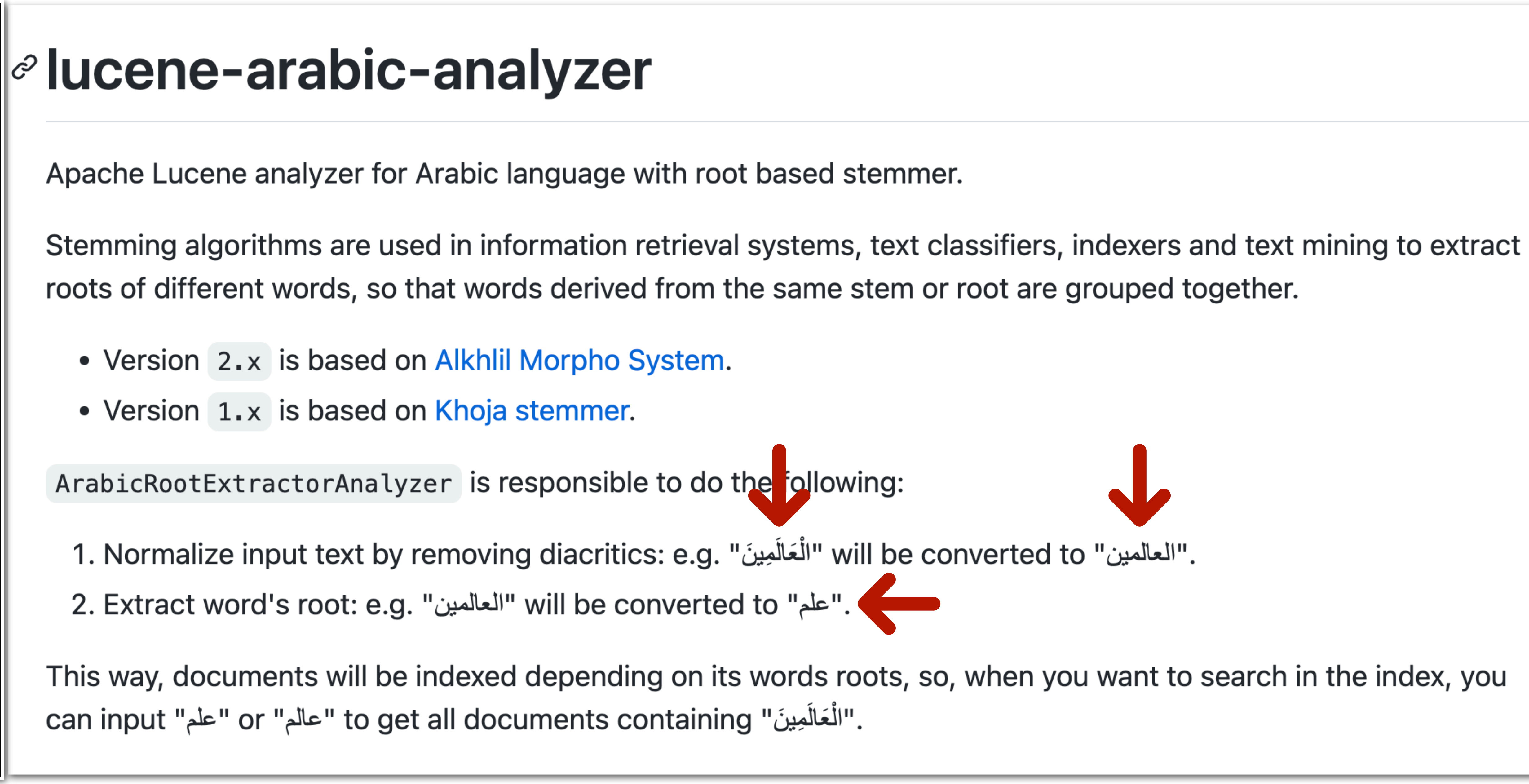
CREATE INDEX INDEX_NAME ON TABLE_NAME(FIRST_NAME, LAST_NAME);



```
SELECT * FROM restaurants WHERE latitude  > 51.4946 AND latitude  < 51.5079
                                AND longitude > -0.1162 AND longitude < -0.1004;
```



TRIE



lucene-arabic-analyzer

Apache Lucene analyzer for Arabic language with root based stemmer.

Stemming algorithms are used in information retrieval systems, text classifiers, indexers and text mining to extract roots of different words, so that words derived from the same stem or root are grouped together.

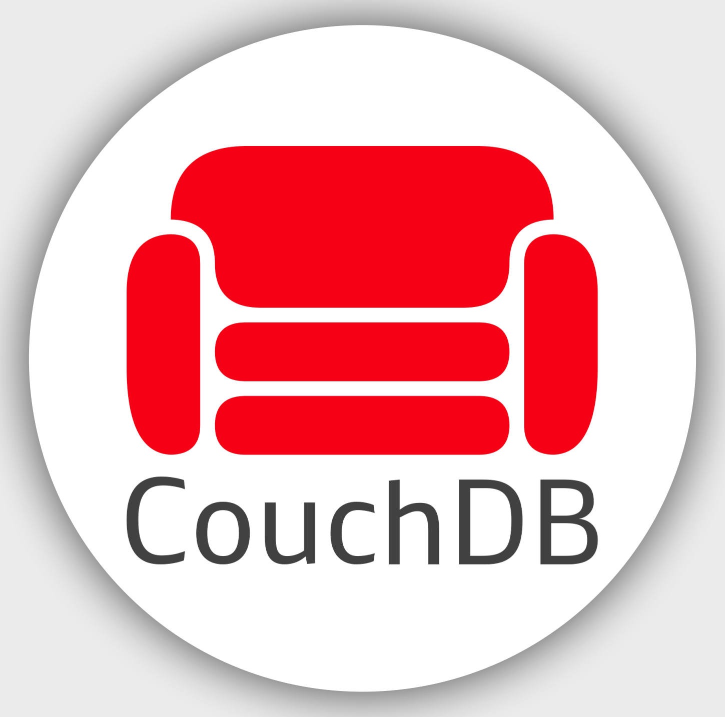
- Version `2.x` is based on Alkhlil Morpho System.
- Version `1.x` is based on Khoja stemmer.

`ArabicRootExtractorAnalyzer` is responsible to do the following:

1. Normalize input text by removing diacritics: e.g. "أَلْعَالَمِينَ" will be converted to "العالمين".
2. Extract word's root: e.g. "العالمين" will be converted to "علم".

This way, documents will be indexed depending on its words roots, so, when you want to search in the index, you can input "علم" or "عالم" to get all documents containing "أَلْعَالَمِينَ".

IN-MEMORY DATABASES



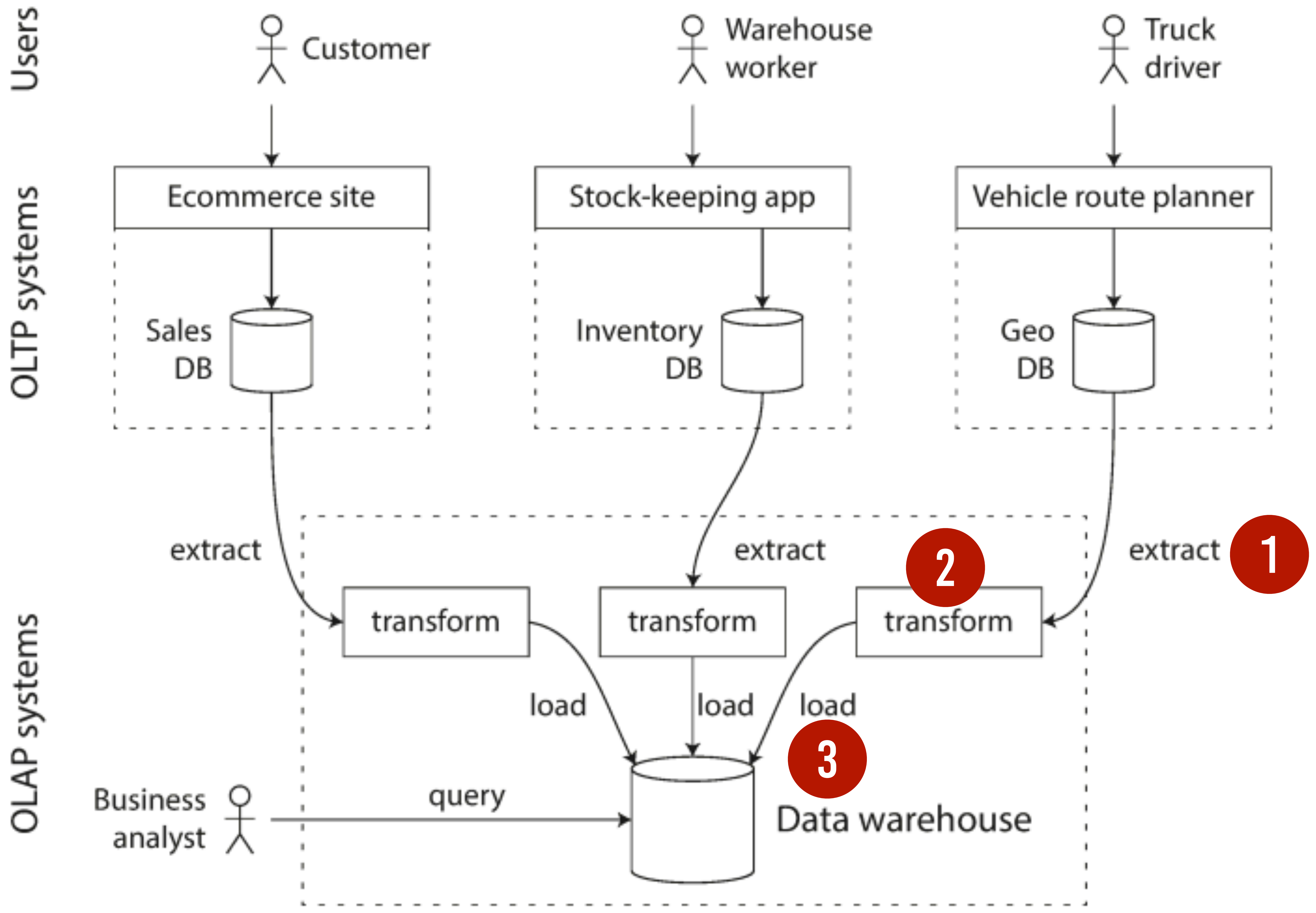
OLTP

ONLINE TRANSACTIONAL PROCESSING

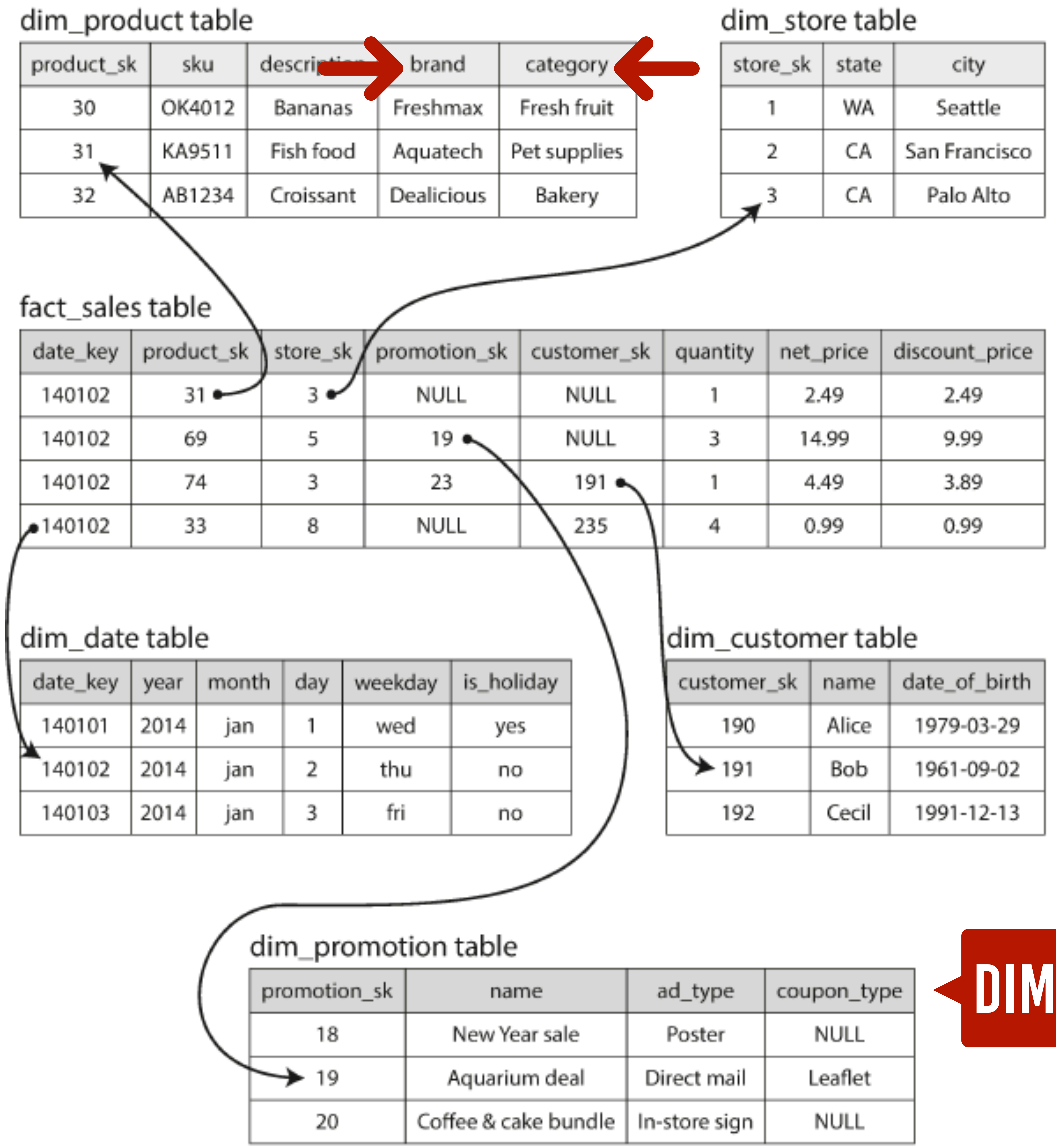
OLAP

ONLINE ANALYTICAL PROCESSING

THE DATA WAREHOUSE



STARS SCHEMA VS SNOWFLAKES SCHEMA



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

DIMENSION TABLE

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

FACT TABLE

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

DIMENSION TABLE

dim_promotion table

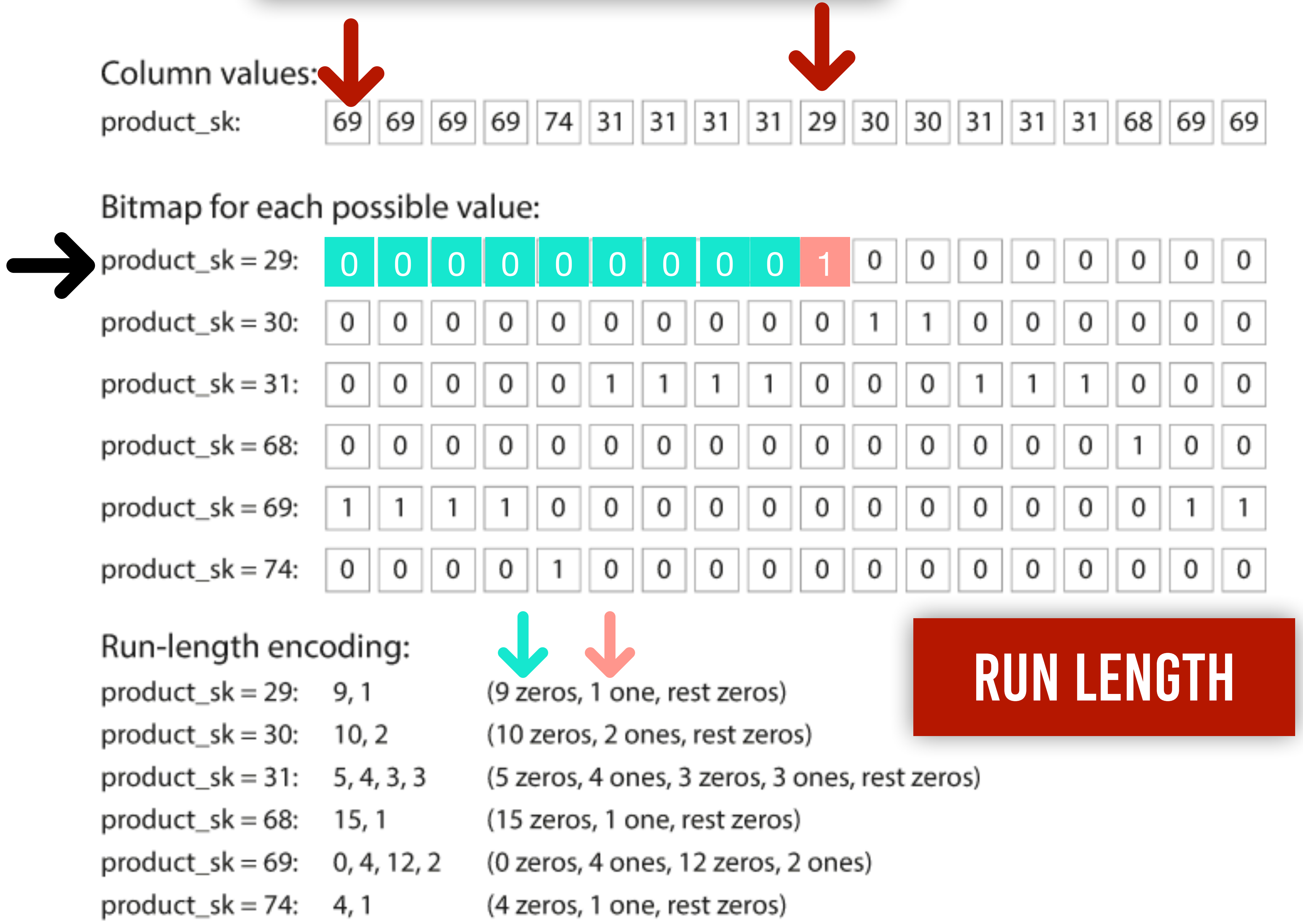
promotion_sk	name	ad_type	coupon_type
18	New Year sale	Poster	NULL
19	Aquarium deal	Direct mail	Leaflet
20	Coffee & cake bundle	In-store sign	NULL

DIMENSION TABLE

COLUMN ORIENTED STORAGE

```sql
SELECT
  dim_date.weekday, dim_product.category,
  SUM(fact_sales.quantity) AS quantity_sold
FROM fact_sales
  JOIN dim_date    ON fact_sales.date_key   = dim_date.date_key
  JOIN dim_product ON fact_sales.product_sk = dim_product.product_sk
WHERE
  dim_date.year = 2013 AND
  dim_product.category IN ('Fresh fruit', 'Candy')
GROUP BY
  dim_date.weekday, dim_product.category;
```

BITMAP ENCODING



Column values:

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

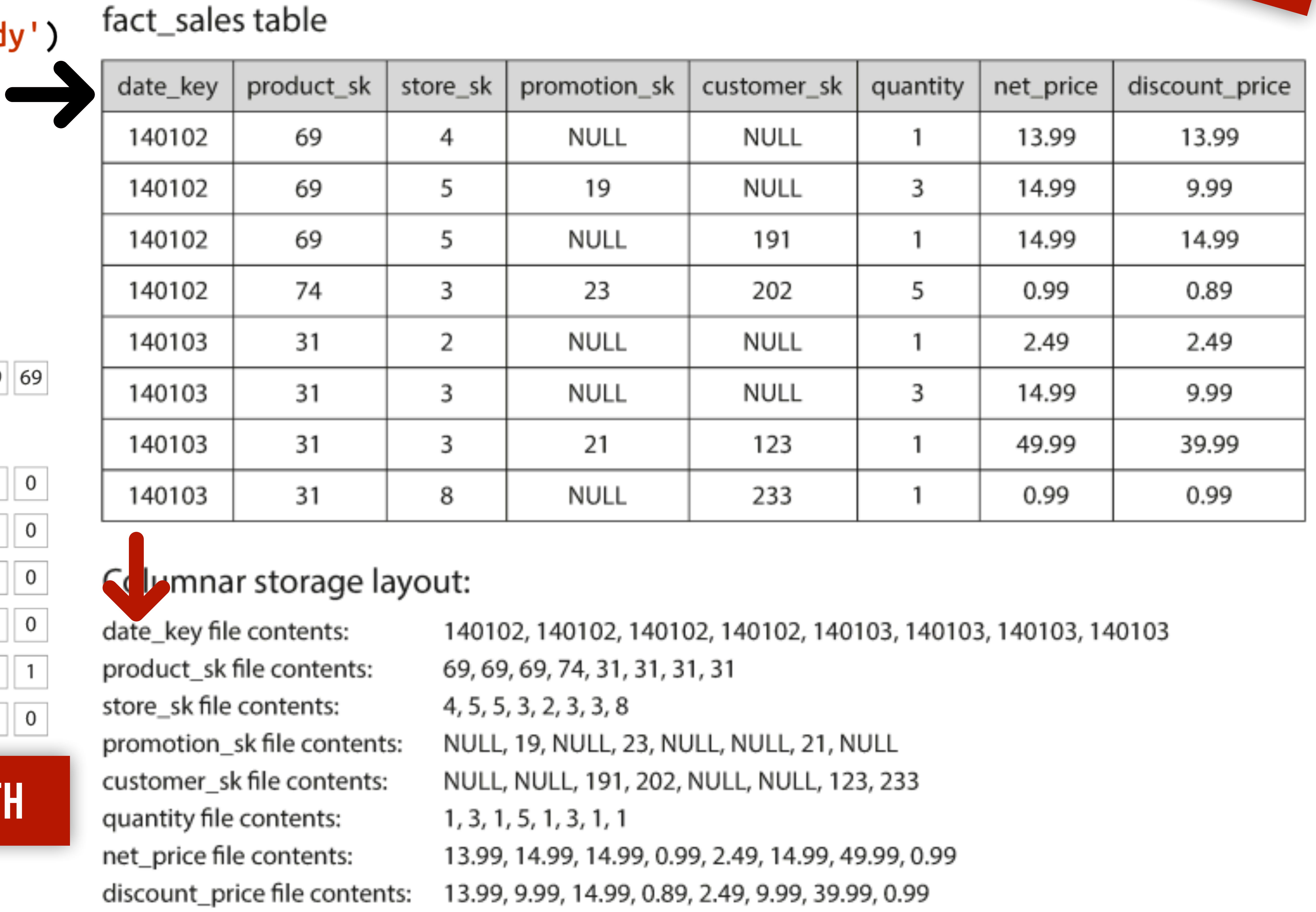
Bitmap for each possible value:

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)

RUN LENGTH



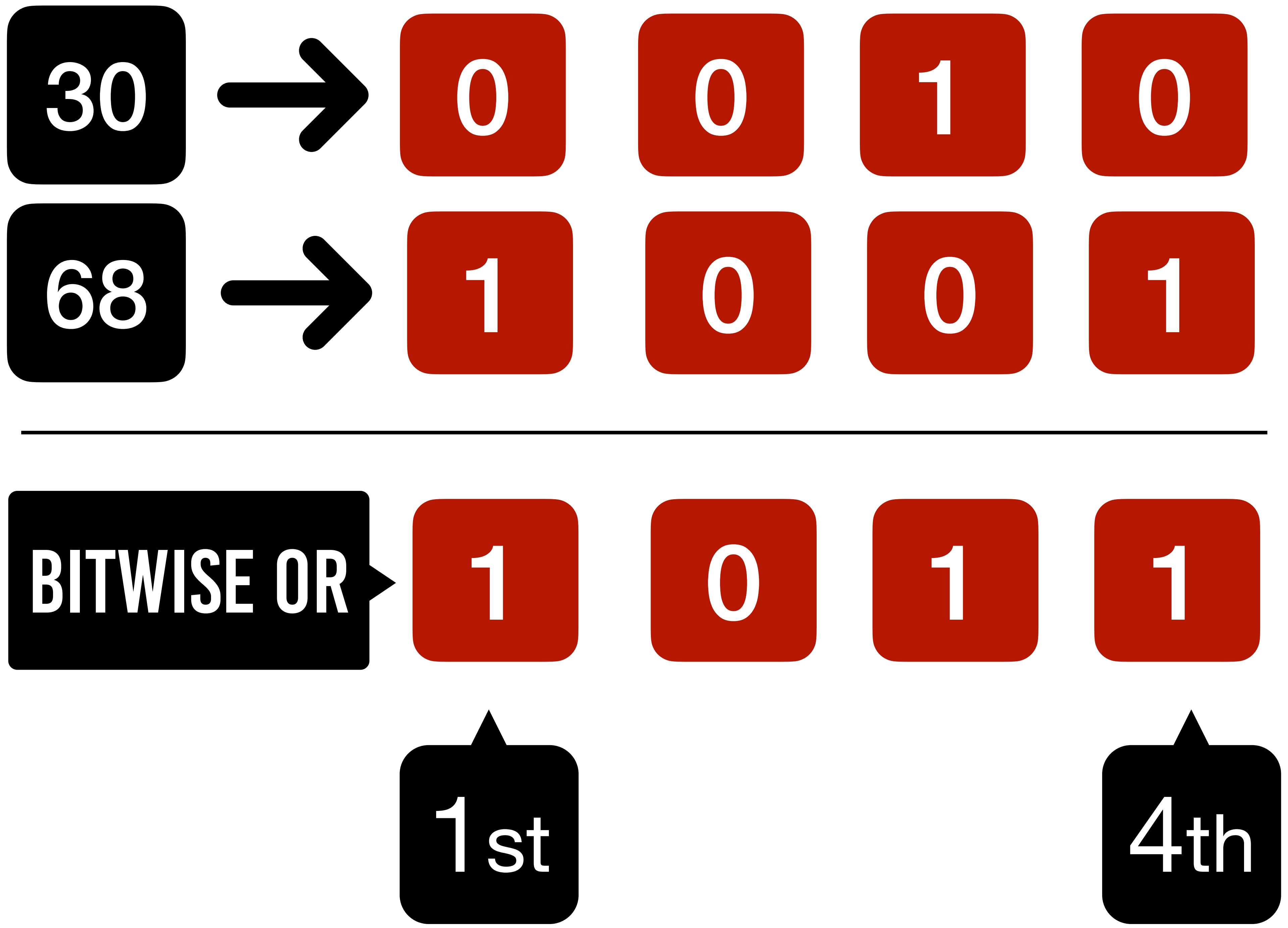
fact_sales table

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

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



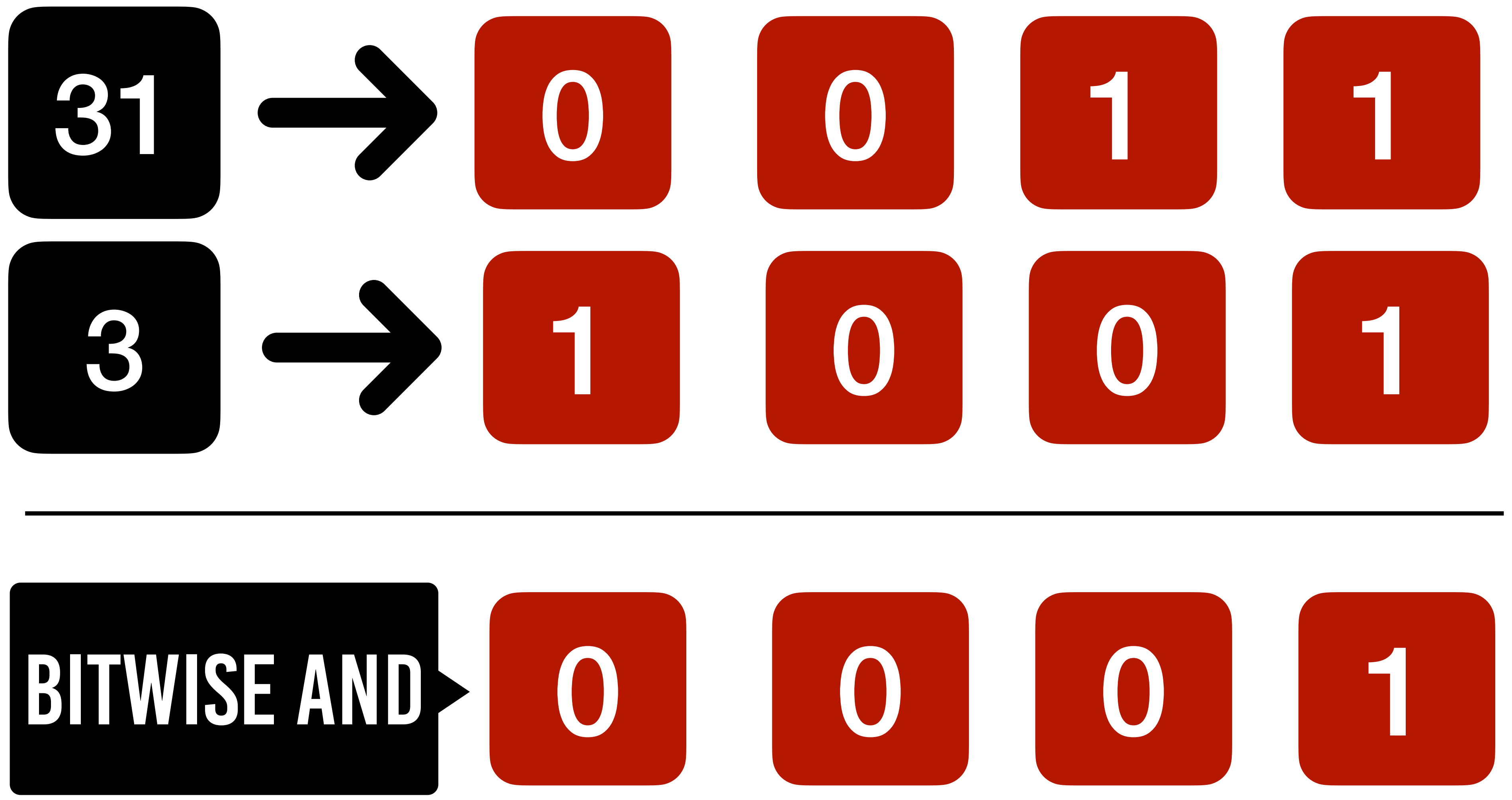
30 → 0 0 1 0

68 → 1 0 0 1

BITWISE OR 1 0 1 1

1st 4th

```
WHERE product_sk = 31 AND store_sk = 3
```



31 → 0 0 1 1

3 → 1 0 0 1

BITWISE AND 0 0 0 1

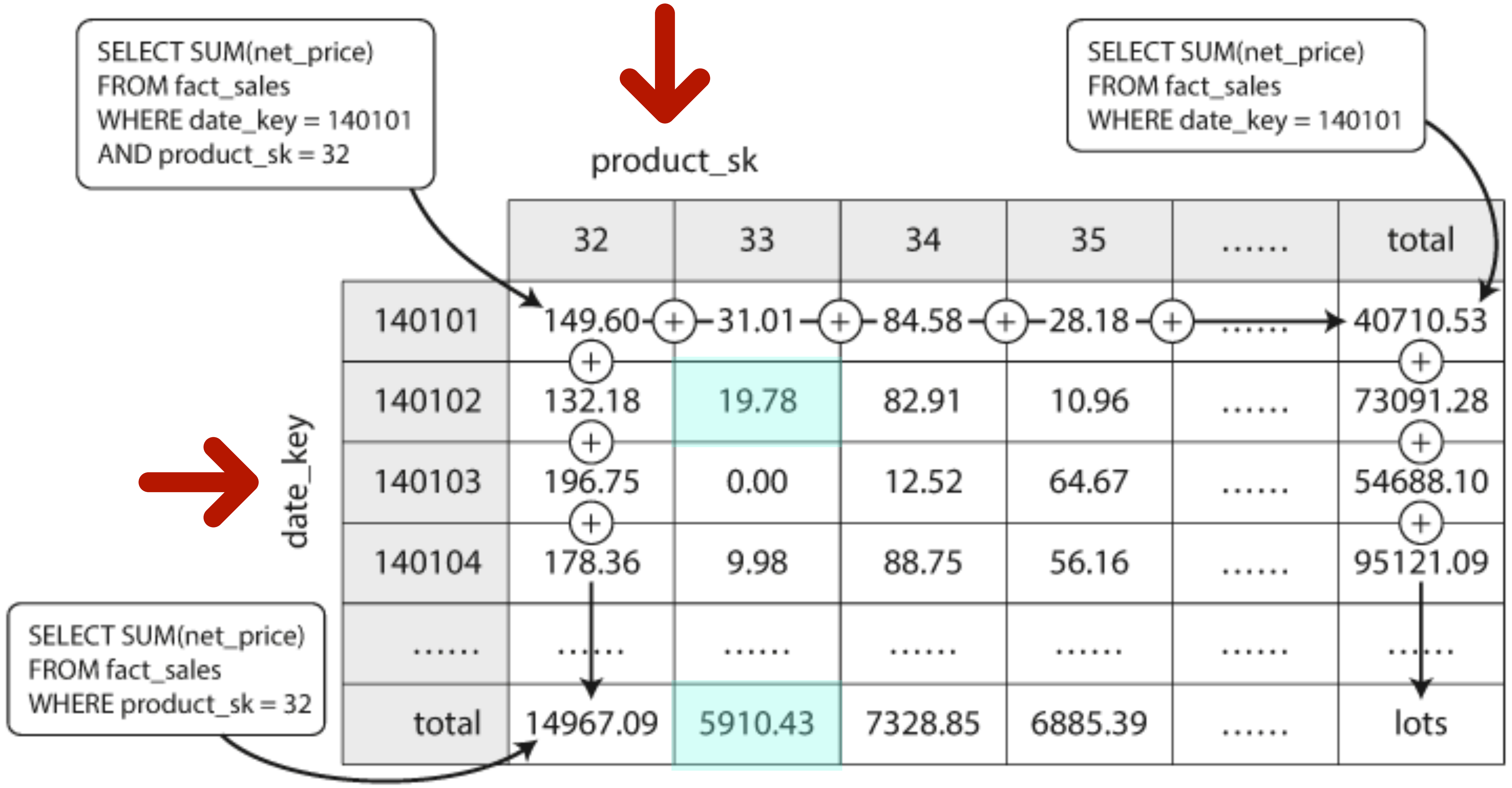
AGGREGATE QUERIES

AVG / SUM / COUNT

MATERIALIZED VIEWS

RESULT COPY ON DISK

DATA CUBE / OLAP CUBE



```
SELECT SUM(net_price)
FROM fact_sales
WHERE date_key = 140101
AND product_sk = 32
```

```
SELECT SUM(net_price)
FROM fact_sales
WHERE date_key = 140101
```

```
SELECT SUM(net_price)
FROM fact_sales
WHERE product_sk = 32
```

	32	33	34	35	......	total
140101	149.60	31.01	84.58	28.18	......	40710.53
140102	132.18	19.78	82.91	10.96	......	73091.28
140103	196.75	0.00	12.52	64.67	......	54688.10
140104	178.36	9.98	88.75	56.16	......	95121.09
......	......	......	......	......	......	......
total	14967.09	5910.43	7328.85	6885.39	......	lots