

Large-Scale and Multi-Structured Databases

Column Databases

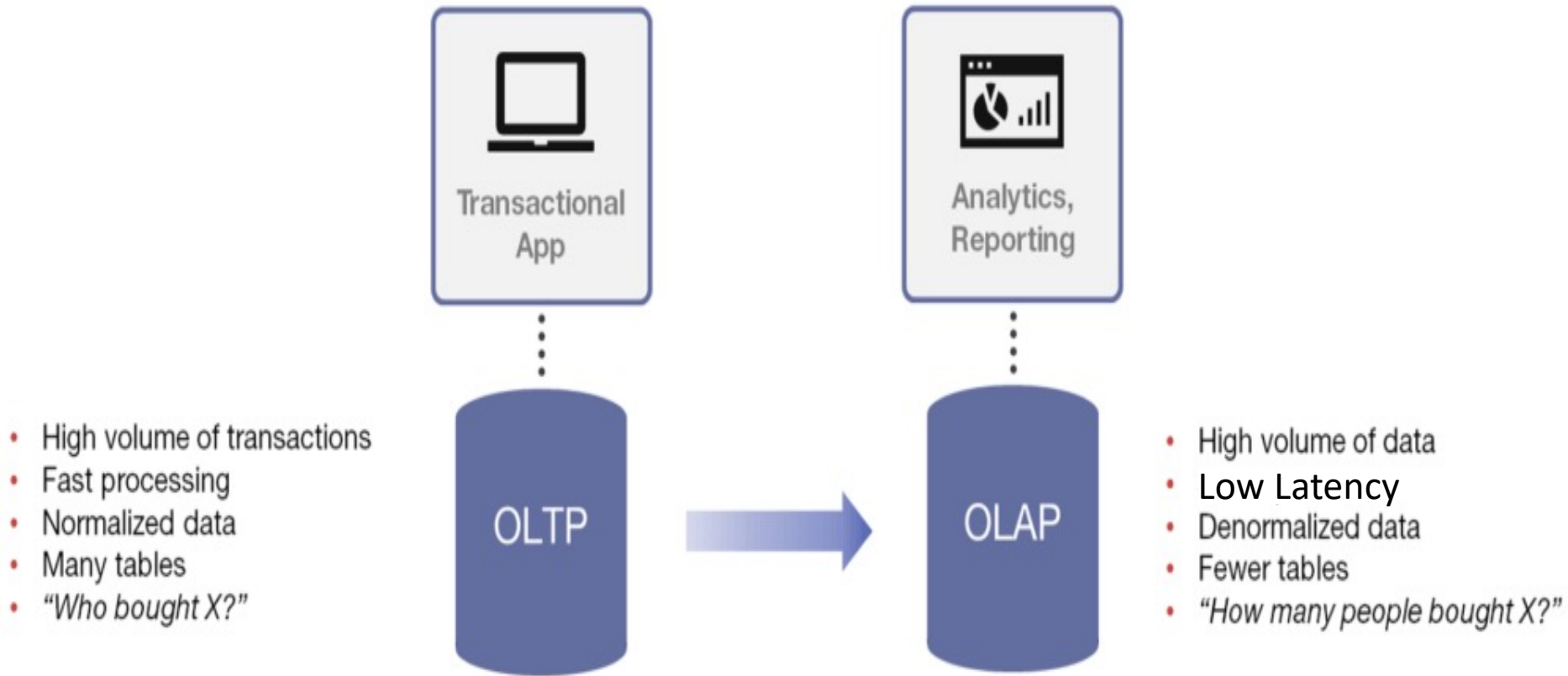
Introduction

Prof. Pietro Ducange

OLTP vs OLAP

- **OLTP** - Online Transaction Processing: software architectures oriented towards **handling ACID transactions**.
- **OLAP** - Online Analytics Processing: software architectures oriented towards **interactive** and **fast analysis** of data. Typical of **Business Intelligence** Software.

OLTP vs OLAP



Row Data Organization

Since the beginning of *digital files*, the data of each record were physically organized in *rows*.

OLTP processing is mainly oriented towards handling *one record* at a time processing.

When the *first relational databases* were designed, the world was mainly experiencing the OLTP era.

The *record-oriented* workflow handled by the first relational databases and *the row-oriented* physical structure of early digital files provided *good performance*.

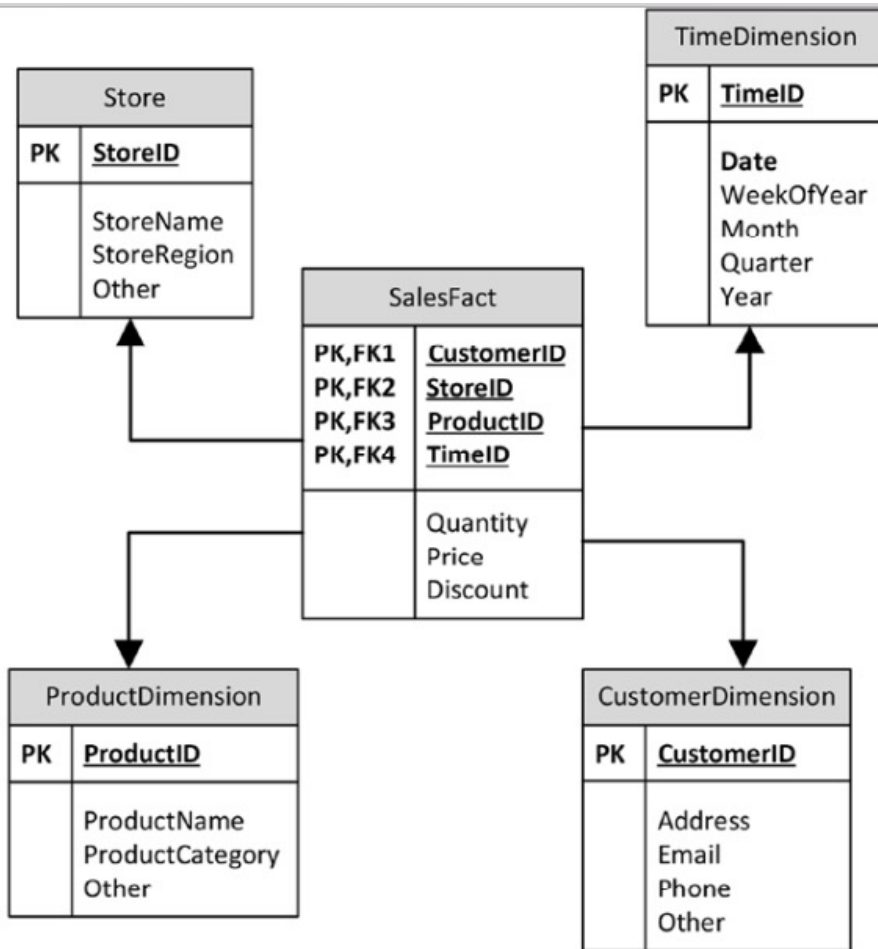
CRUD and Queries

In the ***record-based processing era*** (up to the end of the 80s of the last century), CRUD operations (Create, Read, Update, Delete) were the most time-critical ones.

Reporting programs were typically iterated through entire tables and were run in a ***background*** batch mode. The ***time response*** to the queries was ***not*** a critical ***issue***.

When ***Business Intelligence*** software started to spread, ***OLAP*** processing assumed ***a big relevance***, thus the ***time response*** to queries became a ***critical issue*** to appropriately handle.

Star Schemas in Data Warehouse



It is a *relational solution* where central *large fact* tables are associated with *numerous smaller dimension* tables.

Aggregate *queries* could execute *quickly*.

Star Schemes do *not* represent a fully *normalized* relational model of data (redundancies are often accepted, information sometimes depend partly on the primary key).

Image extracted from: "Guy Harrison, Next Generation Databases, Apress, 2015"

Star Schemas: Main Drawbacks

- ***Data processing*** in data warehouses remained severely ***CPU*** and ***IO intensive***.
- The ***data volumes*** grew up along with ***user demands*** for ***interactive*** response times.
- In general, around mid 90s of the last century, the ***discontent*** with traditional data warehousing performance ***increased***.

The Columnar Storage

When processing data for making *analytics*, in most of cases, we are not interested in retrieving *all the information* of each single records.

Indeed, we are interested, for example, in retrieving the values of *one attribute* of a *set of records* (for making trend charts, or calculating statistics).

When dealing with *row-based* storage of records, we have to *access* to *all the attributes* of the considered set for retrieving just the values of one attribute.

If all the values of an attribute are *grouped* together on the *disk* (or in the block of a disk), the task of retrieving the values of one attribute will be *faster* than a row-based storage of records.

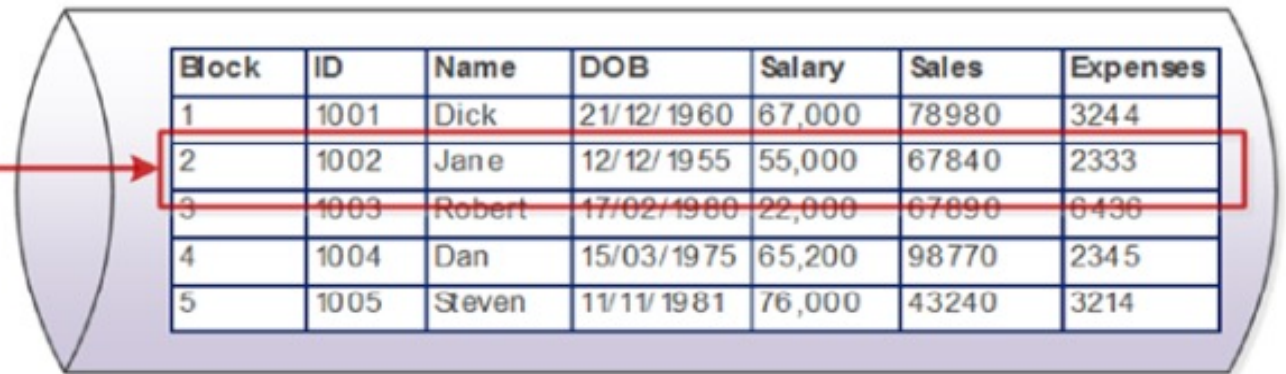
Rows vs Columns

Row Storage

Last Name	First Name	E-mail	Phone #	Street Address

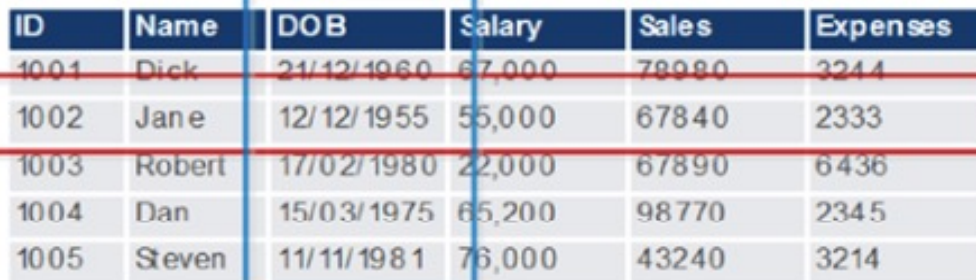
Columnar Storage

Last Name	First Name	E-mail	Phone #	Street Address



Block	ID	Name	DOB	Salary	Sales	Expenses
1	1001	Dick	21/12/1960	67,000	78980	3244
2	1002	Jane	12/12/1955	55,000	67840	2333
3	1003	Robert	17/02/1980	22,000	67890	6436
4	1004	Dan	15/03/1975	65,200	98770	2345
5	1005	Steven	11/11/1981	76,000	43240	3214

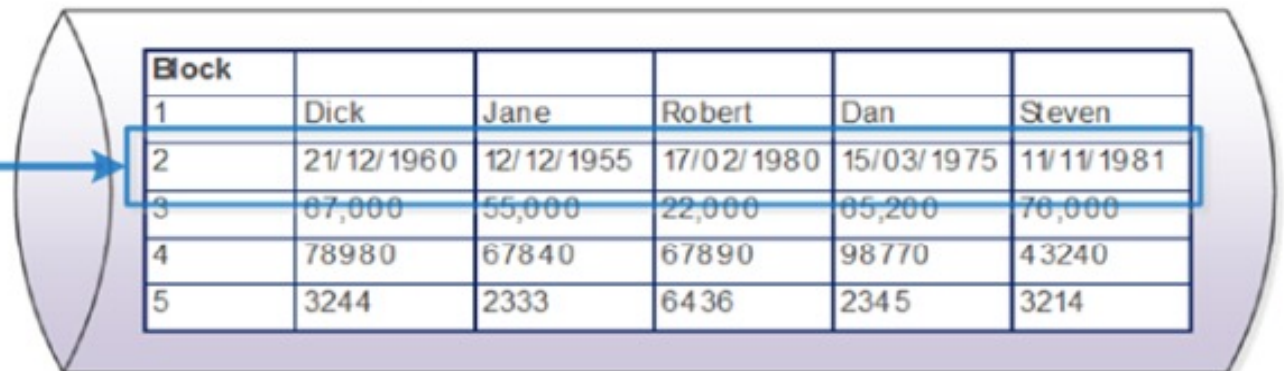
Row-oriented storage



ID	Name	DOB	Salary	Sales	Expenses
1001	Dick	21/12/1960	67,000	78980	3244
1002	Jane	12/12/1955	55,000	67840	2333
1003	Robert	17/02/1980	22,000	67890	6436
1004	Dan	15/03/1975	65,200	98770	2345
1005	Steven	11/11/1981	76,000	43240	3214

Tabular data

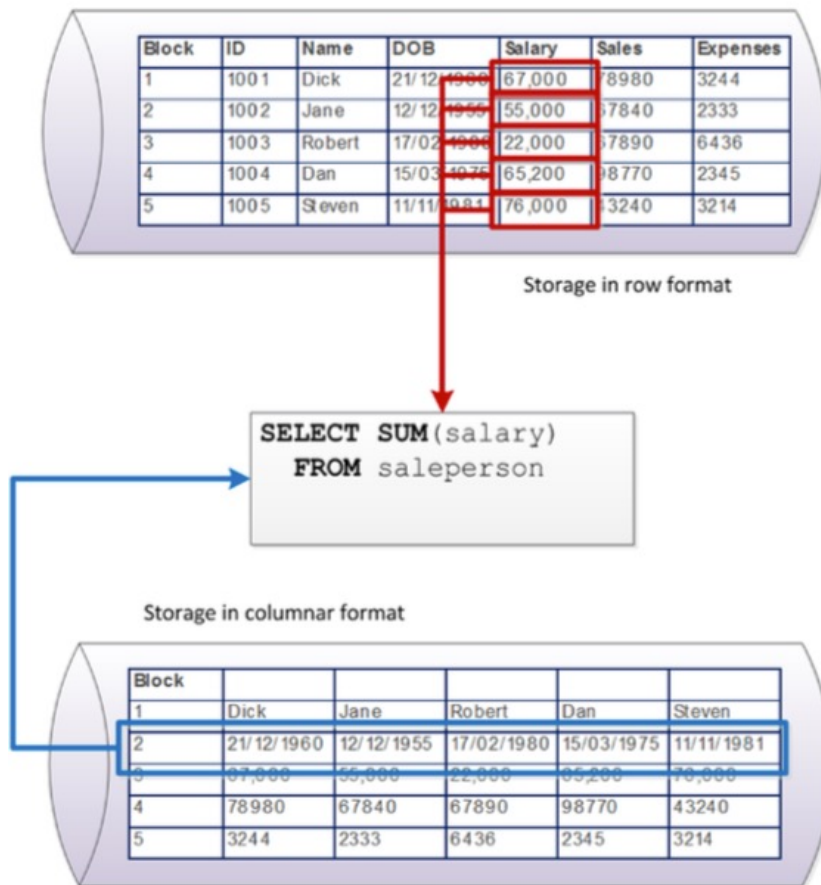
Columnar storage



Block	Dick	Jane	Robert	Dan	Steven
1					
2	21/12/1960	12/12/1955	17/02/1980	15/03/1975	11/11/1981
3	67,000	55,000	22,000	65,200	76,000
4	78980	67840	67890	98770	43240
5	3244	2333	6436	2345	3214

Image extracted from: "Guy Harrison, Next Generation Databases, Apress, 2015"

An Example



If data are stored by **rows**, in order to retrieve the sum of salaries we must scan **five** blocks.

In the case of **column** store, a **single block access** is enough.

In general, queries that work across **multiple rows** are significantly accelerated in a columnar database.

Image extracted from: "Guy Harrison, Next Generation Databases, Apress, 2015"

Columnar Compression

Data compression algorithms basically *remove redundancy* within data values.

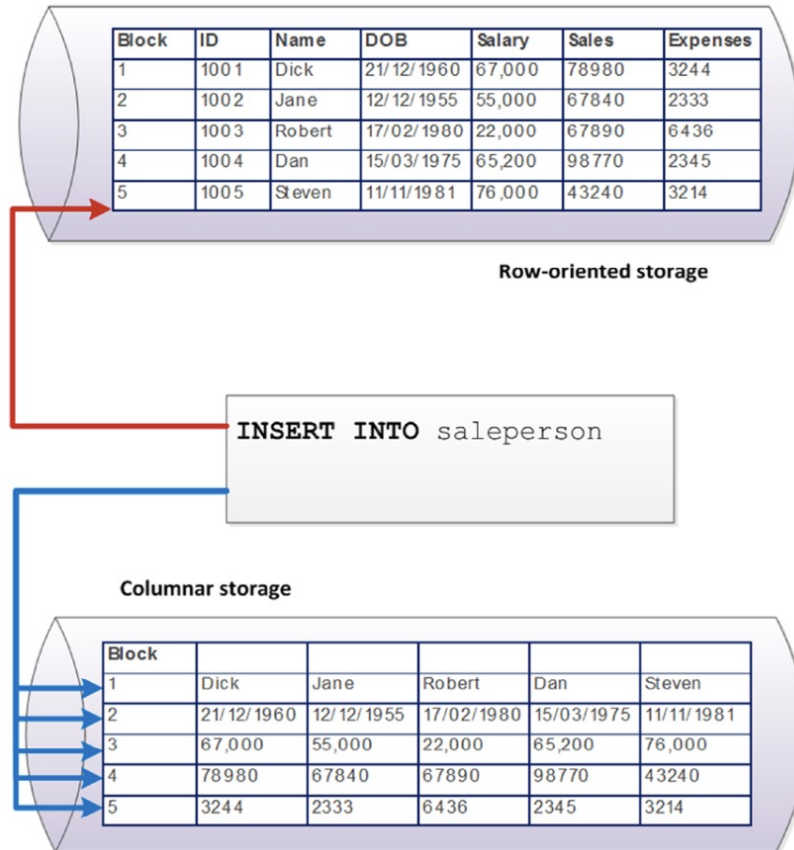
The higher the redundancy, the higher the compression ratios.

To find redundancy, data compressions algorithms usually try to *work* on *localized* subsets of the data.

If data are stored in a column database, very *high compression ratio* can be achieved with very *low computational overheads*.

As an example, often in a columnar database data are stored in a *sorted order*. In this case, very high compression ratios can be achieved simply by representing each column value as a “*delta*” from the preceding column value.

Single Row Operations Penalty



Column databases perform *poorly* during *single-row modifications*.

The *overhead* needed for *reading* rows, can be *partly reduced* by caching and multicolumn projections (storing multiple columns together on disk).

In general, handling a row means accessing to more than once blocks of the disk.

Image extracted from: "Guy Harrison, Next Generation Databases, Apress, 2015"

Delta Store

Problem: The basic column storage architecture is *unable* to cope with *constant* stream of *row-level modifications*.

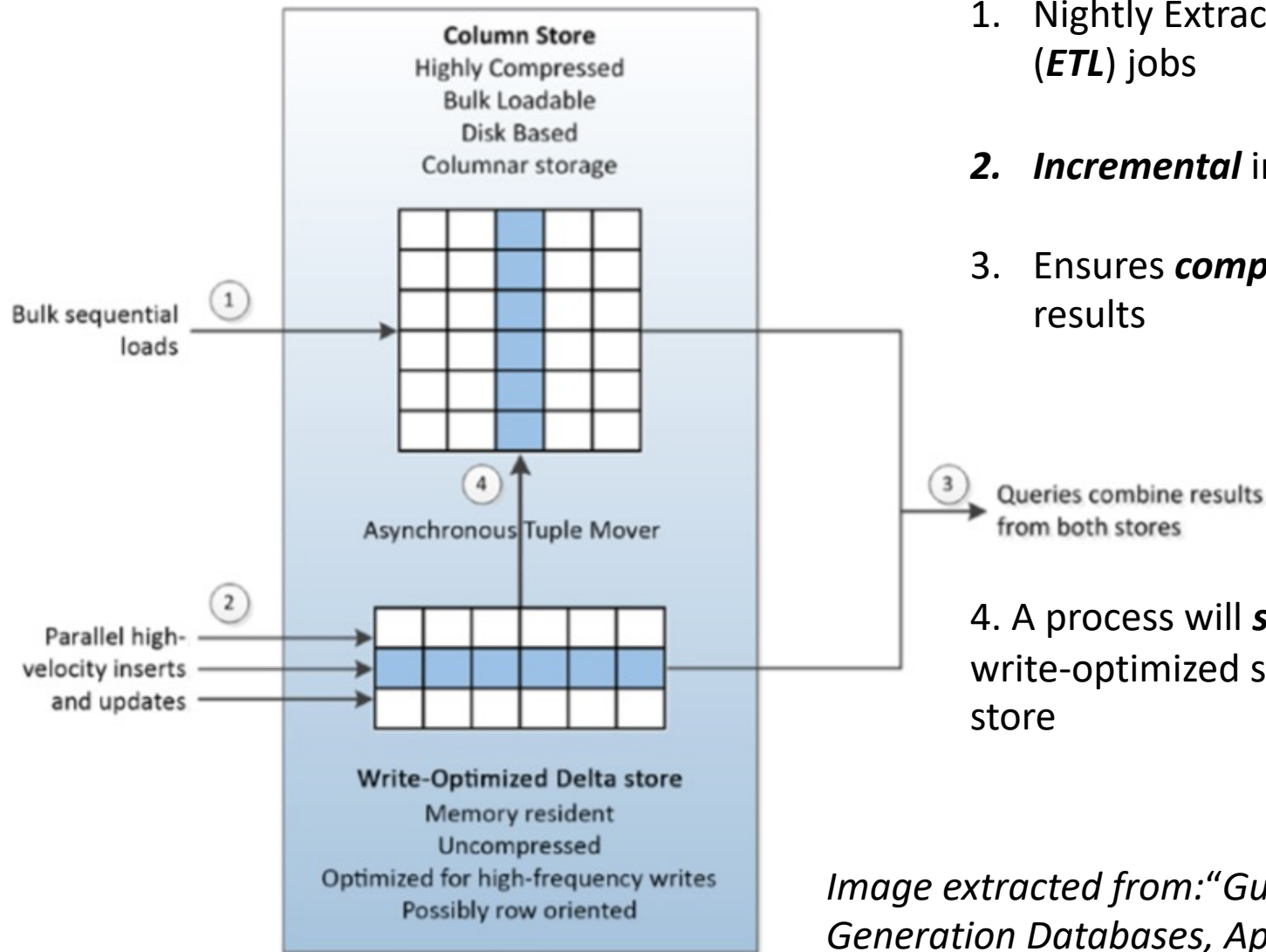
These modifications are needed whenever *data warehouse* have to provide real-time “*up-to-the-minute*” information .

Solution: *Delta store* is an area of the database, resident in *in-memory uncompressed* and optimized for high-frequency data modifications.

Data in the delta store is *periodically merged* with the main *columnar-oriented* store (often data are *highly compressed*).

Queries may need to *access both* the delta store and the column store in order to return complete and accurate results.

Delta Store: Architecture



1. Nightly Extract, Transform, Load (**ETL**) jobs
2. **Incremental** inserts and updates
3. Ensures **complete** and **consistent** results
4. A process will **shift data** from the write-optimized store to the column store

Image extracted from: "Guy Harrison, Next Generation Databases, Apress, 2015"

Projections

Problem:

Complex queries often need to read ***combinations*** of column data.

Solution:

Some column databases adopts ***projections***.

Projections: combinations of columns that are ***frequently*** accessed together.

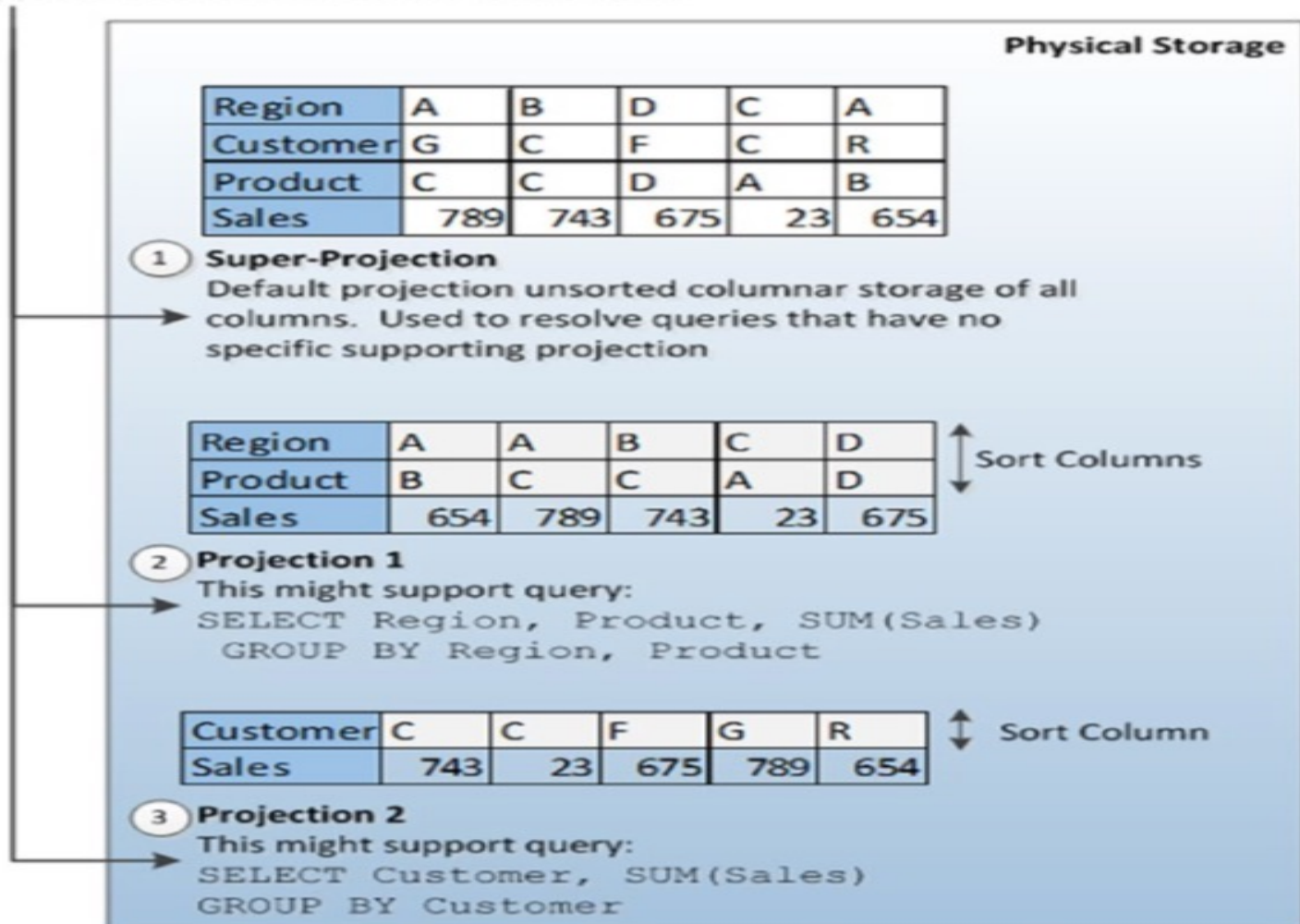
Columns of specific projections are ***stored together*** on the disk.

In the following slide, we show an example of database architecture based on projections.

Region	Customer	Product	Sales
A	G	C	789
B	C	C	743
D	F	D	675
C	C	A	23
A	R	B	654

Logical Table

Table appears to user in relational normal form



Suggested Readings

Chapter 6 of the book “*Guy Harrison, Next Generation Databases, Apress, 2015*”.