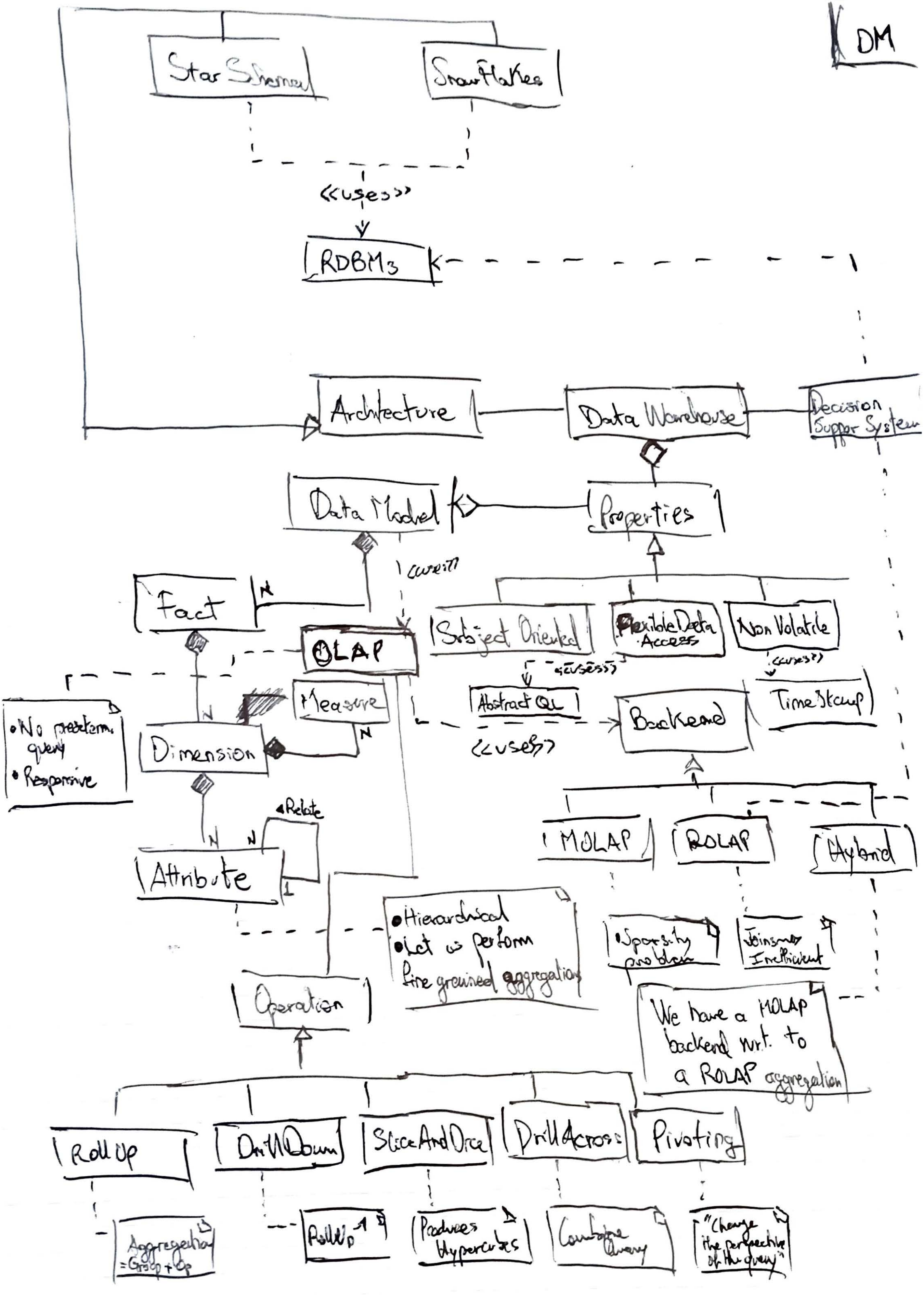
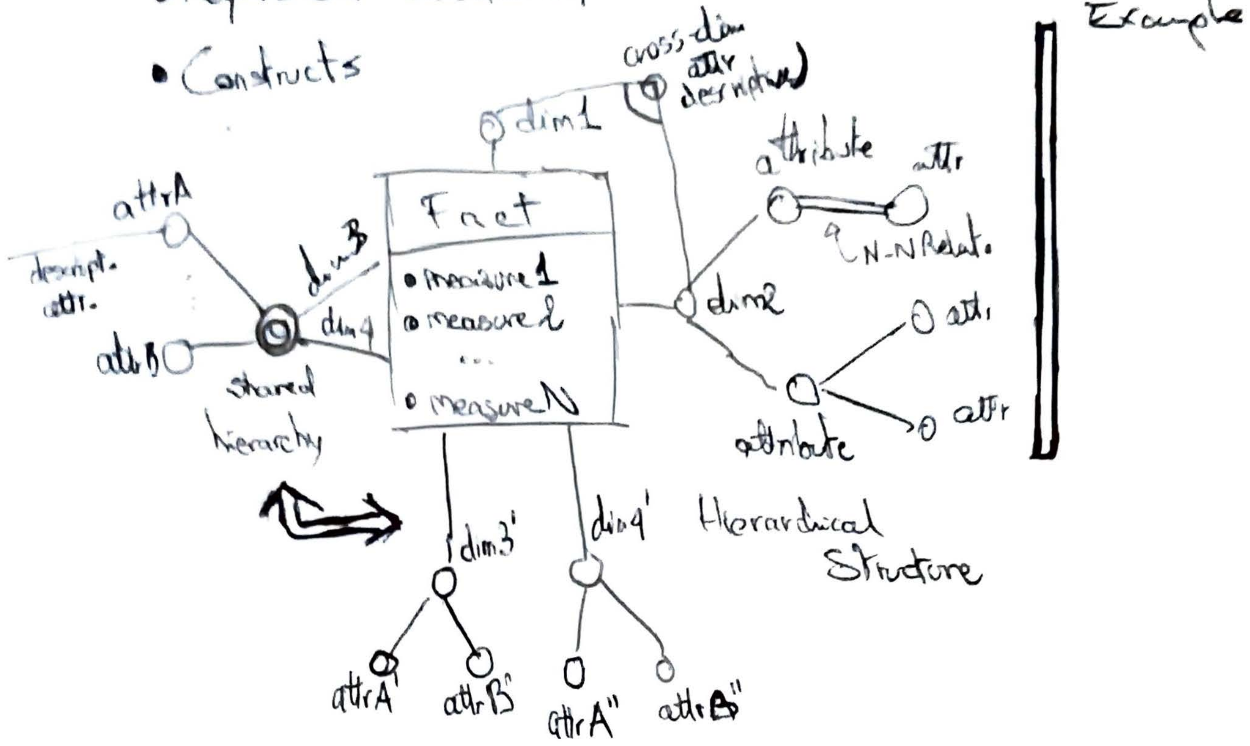

DM
Star Schema
SnowFlakes
<<uses>>
RDBMS
Architecture
Data Warehouse
Decision Suppor System
Data Model
Properties
<<uses>>
Fact
N
OLAP
Subject Oriented
Flexible Data Access
Non Volatile
<<uses>>
<<uses>>
Abstract QL
Backend
Time Stamp
<<uses>>
Measure
N
• No predeterm. query
• Responsive
Dimension
N
Relate
N
N
1
Attribute
MOLAP
ROLAP
Hybrid
• Hierarchical
• Let us perform fine grained aggregation
• Sparsity problem
Joins Inefficient
Operation
We have a MOLAP backend wrt. to a ROLAP aggregation
Roll Up
DrillDown
SliceAndDice
DrillAcross
Pivoting
Aggregation = Group + Op
RollUp^{-1}
Produces Hypercubes
Combine Query
"Change the perspective of the query"

Design: Dimensional Fact Model

- Requirement Elicitation
- Constructs



- Measures are attribute we want to investigate (we want to assign them a value)

- Attributes are useful for aggregation

- Descriptive attributes are "additional values".
- Dimensions are just the root of an hierarchy.

... We can ~~also~~ also have facts that need to be counted only...

NOTE: An instance of a fact is ~~based~~ an element of the set of $(\{dims\} \cup \{attrs\})$

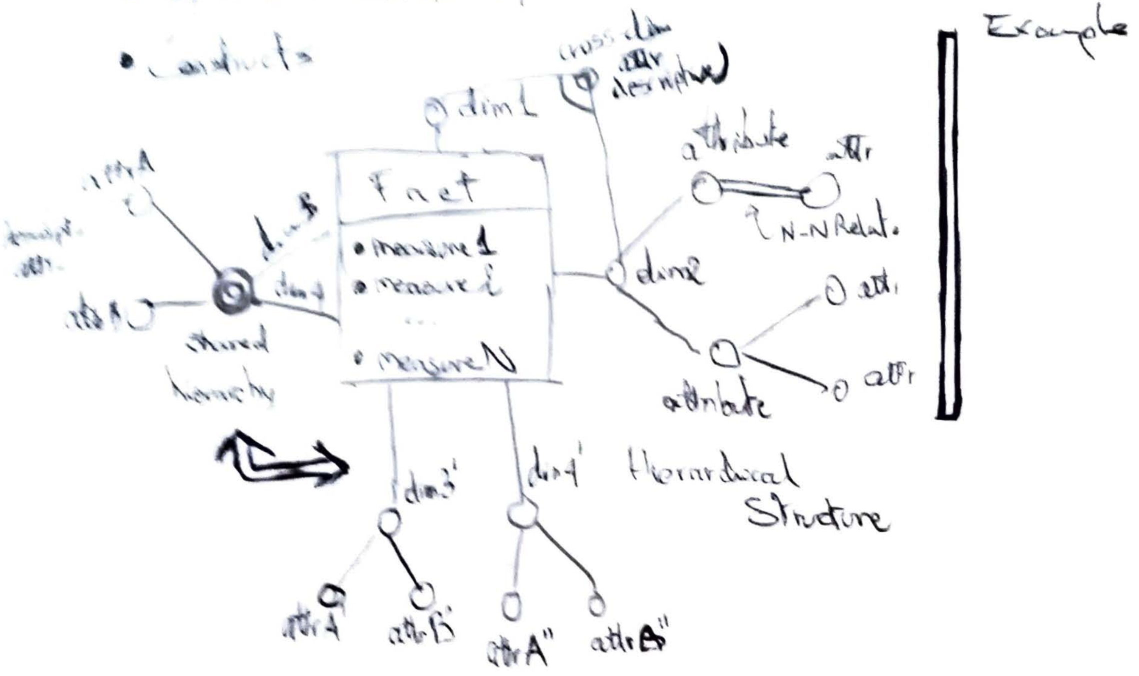
Homework Related Stuff

ROLAP → Star Schema → Fact & Dim Tables → Flatten the hierarchy
~~The~~ $\xrightarrow{\text{SOLUTION*}}$ SnowFlakes → Reduce Redundancy

* We don't need to "snowflake" the entire hierarchy (recursively)

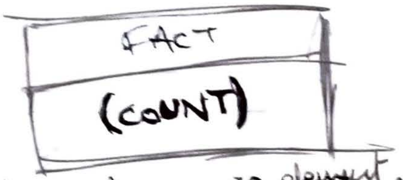
Design: Dimensional Fact Model

- Requirement Elicitation
- Constructs



- Measures are attribute we want to investigate (we want to assign them a value)
- Attributes are useful for aggregation
- Descriptive attributes are "additional values".
- Dimensions are just the root of an hierarchy.

... We can also have facts that need to be counted only...



NOTE: An instance of a fact is ~~based~~ an element of the set of $(\{dims\} \cup \{attrs\})$

Homework Related Stuff

ROLAP ⇝ Star Schema ⇝ Fact & Dim Tables ⇝ Flattern the hierarchy

The $\overset{solution*}{\rightsquigarrow}$ SnowFlakes ⇝ Reduce Redundancy

* We don't need to "snowflake" the entire hierarchy (recursively)

NOTE: Use surrogate Key instead of Business Key (Hint: Autoincrement Primary Key).

NOTE: Snowflaking produces:

⊕ Reduce occupied Space

⊕ Reading based on dimensions is efficient...

⊖... ~~doing it on attributes needs Join~~ but if it doesn't happen it's no Inefficient. (As the case of [illegible])

Views

We can materialise aggregation exploiting materialized view.

Notice that we can only store associative aggregate schema.

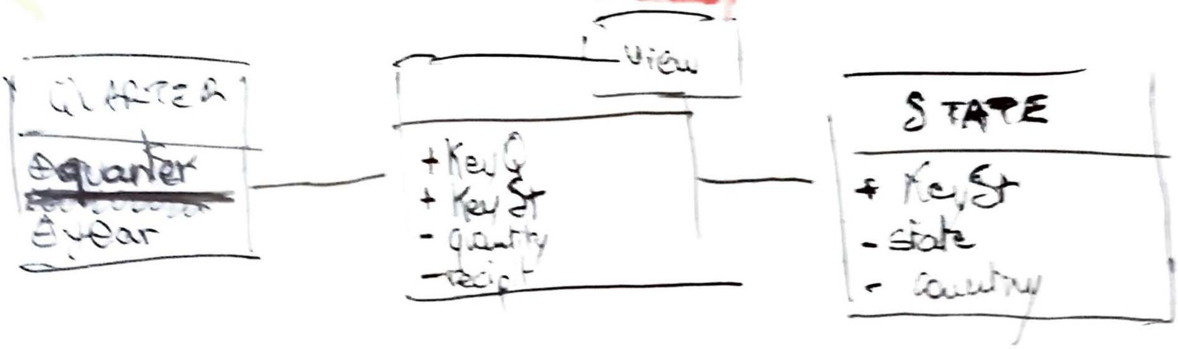
Example

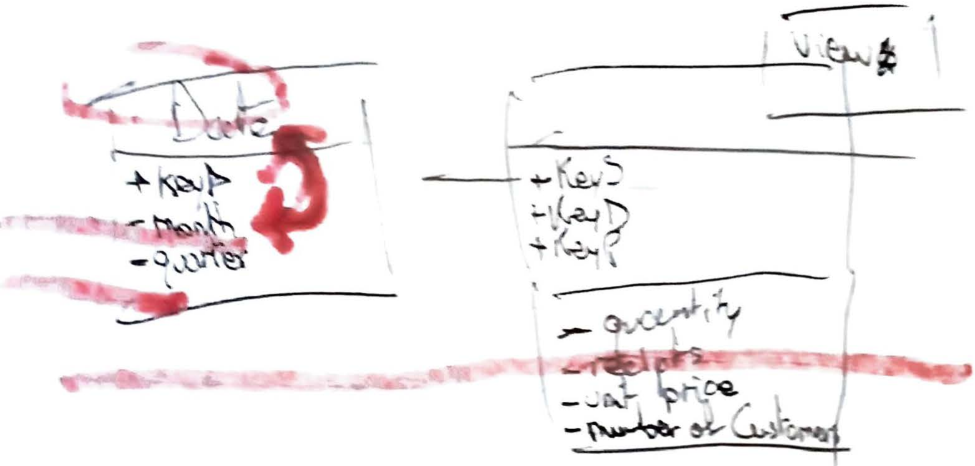
sum(sum, sum) ✓

avg(avg, avg) ✗

We have only a surrogate for them → Derived measures

Scale = 1.26





Costellation Schema: Multiple stars (Remember the next 2 subsections)

Costellation Schema: Multiple stars (Rember the next 2 subsection)

A tipical architectre is 2-views architecture:

1° PRIMARY FACT (∧)

1° SECONDARY FACT

NOTE:

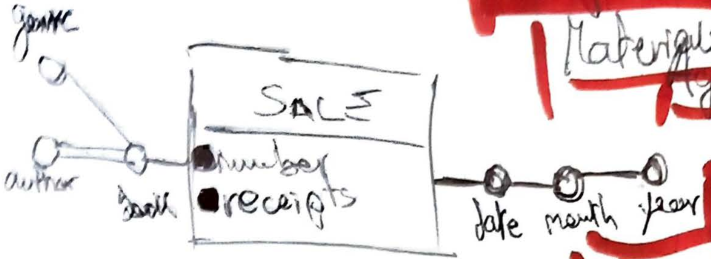
- ~~Note that~~ ~~either~~ snowflackes → shared herarchies → Ease of implementation

[We can start thinking of ...] • ~~Built dimensions~~?

• ~~Bridge table~~: [The manager PoW]

Bridge Table: N-to-N Relationship

We use a so called "bridge table" Materialized Aggregated View of a non associative or associative aggreg. operation



• The query need to ~~[illegible]~~ consider weights to the arcs

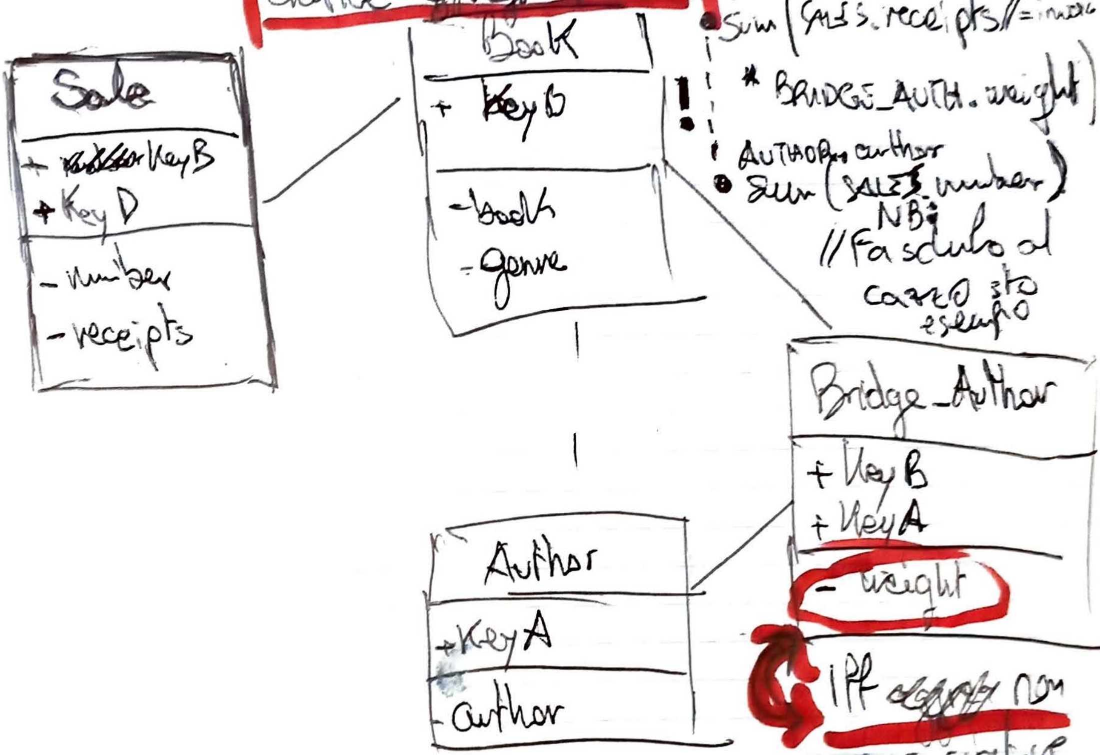
• No weight required

→ Pitfall Example:

• AUTHOR.author,
Sum (SALES.receipts // = [illegible]
* BRIDGE_AUTH.weight)

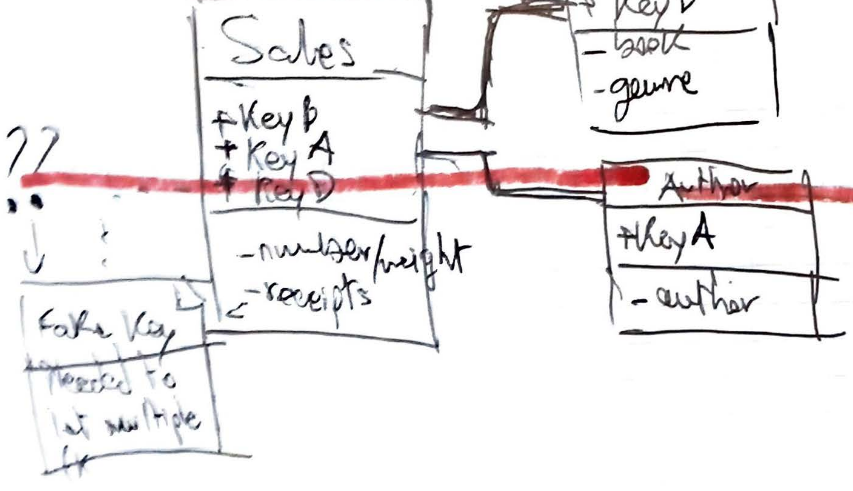
!

• AUTHOR.author
Sum (SALES.number)
NB: // Fa schifo al cazzo sto esempio



iff ~~aggr~~ non associative

Pushdown Solution: N-to-N relationship for non associativity op.



Degenerate Dimensions

byte_len(attr_dims) >> byte_len(surrogate_key)

=> ~~Dimension tables degenerate to~~ THE attributes BECOMES dimensions

JUNK DIM. ~~The attribute gets promoted to a dimension.~~

~~Dimensions get degenerate in (unique) attributes~~

=> Less JOIN operation required (I'm taking it for granted)