# Postgres as BI platform

Andy Fefelov

mastery.pro

# Agenda

- Problem statement
- Open source solution
- ROLAP
- Our architecture review
- Postgres features suitable for BI
  - ETL vs ELT (stage-nds-ddm)
  - Column data storage
  - Configuration
  - Special features
- Pros and cons of out solution

# Problem statement

- Our customer is one of the largest pharmacy supply chain group in Ireland
- 4 types of dispensary software
- 250 pharmacies
- To be analyzed:
  - Orders
  - Scripts (prescription, recipe)
  - Claims
- Goals to be achieved:
  - Purchasing policy optimization
  - Marketing killing feature

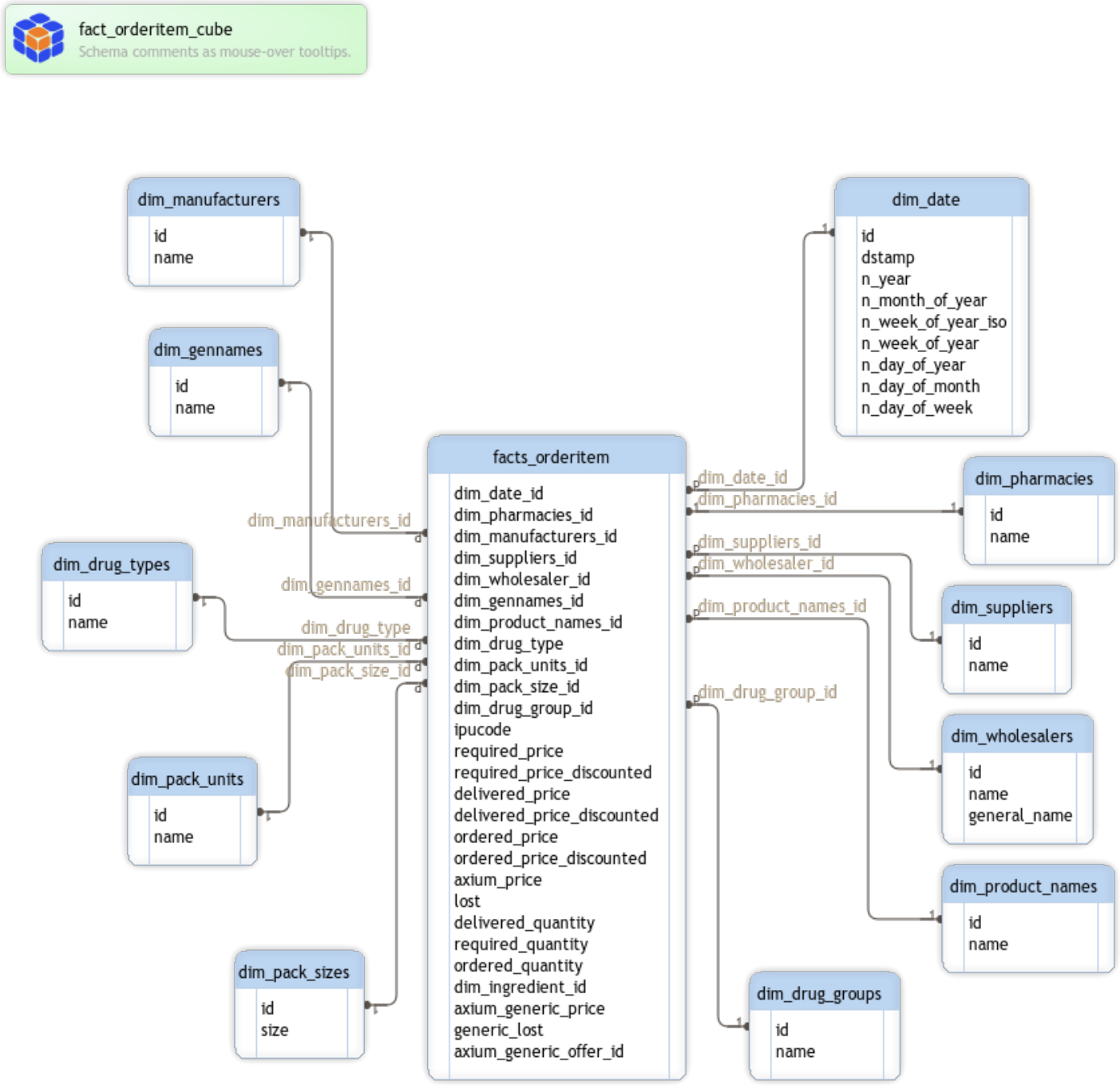
# Open source

- SpagoBI
- Pentaho
- Mondrian
- Saiku
- Cubes (databrewery)

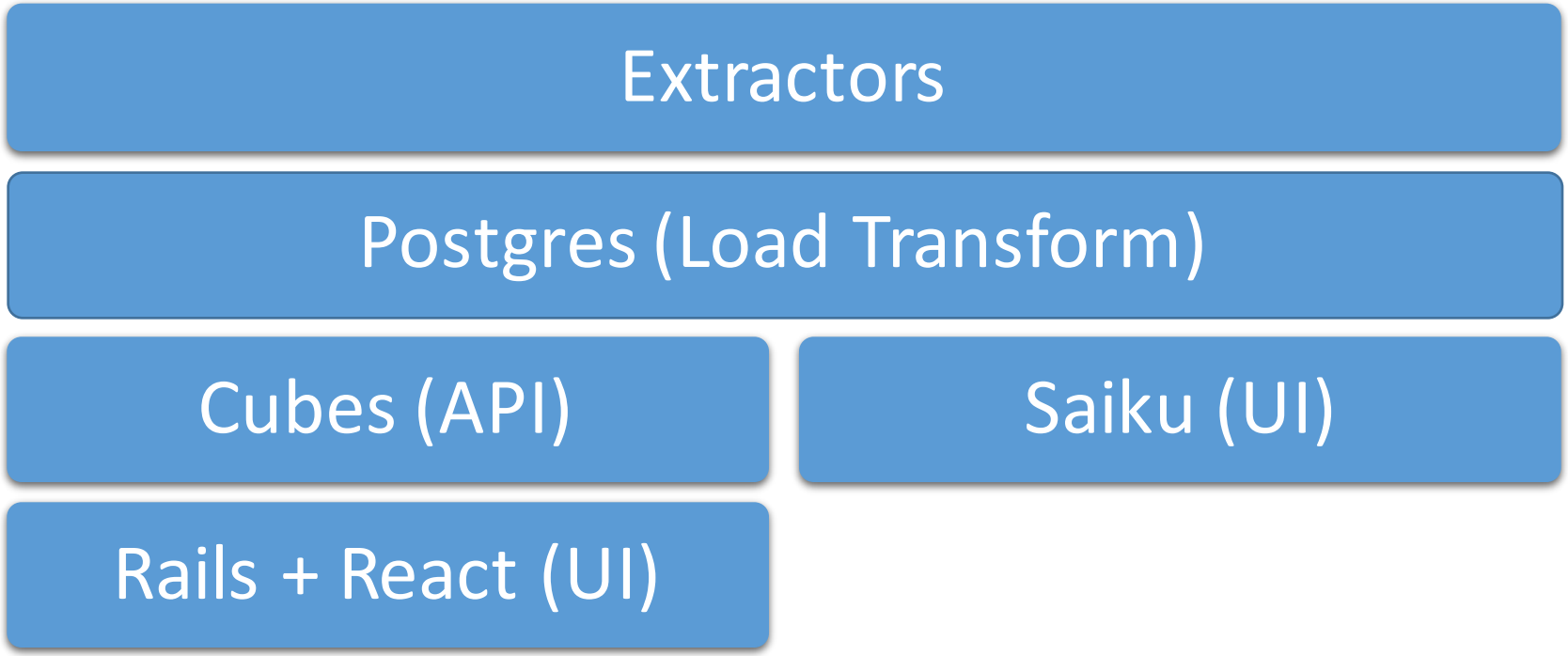
# ROLAP (R-ROLAP)

- Star scheme
  - Facts
  - Dimensions
  - Measures
- No pre-calculated aggregates
- SSD
- Column storage
- ???
- Profit!

# ROLAP

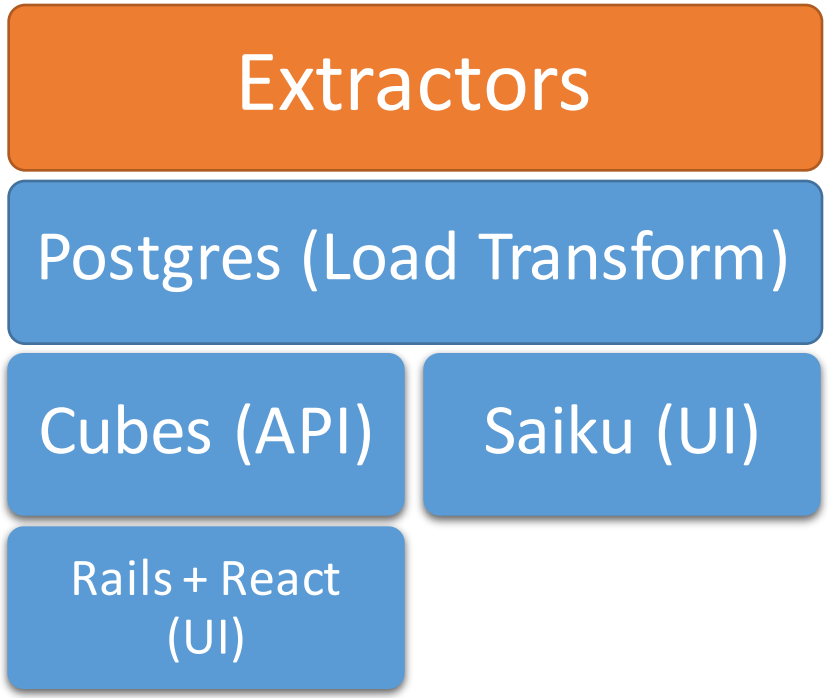

fact_orderitem_cube
Schema comments as mouse-over tooltips.
dim_manufacturers
id
name
dim_gennames
id
name
dim_drug_types
id
name
dim_pack_units
id
name
dim_pack_sizes
id
size
dim_manufacturers_id
dim_gennames_id
dim_drug_type
dim_pack_units_id
dim_pack_size_id
facts_orderitem
dim_date_id
dim_pharmacies_id
dim_manufacturers_id
dim_suppliers_id
dim_wholesaler_id
dim_gennames_id
dim_product_names_id
dim_drug_type
dim_pack_units_id
dim_pack_size_id
dim_drug_group_id
ipucode
required_price
required_price_discounted
delivered_price
delivered_price_discounted
ordered_price
ordered_price_discounted
axium_price
lost
delivered_quantity
required_quantity
ordered_quantity
dim_ingredient_id
axium_generic_price
generic_lost
axium_generic_offer_id
dim_date_id
dim_pharmacies_id
dim_suppliers_id
dim_wholesaler_id
dim_product_names_id
dim_drug_group_id
dim_date
id
dstamp
n_year
n_month_of_year
n_week_of_year_iso
n_week_of_year
n_day_of_year
n_day_of_month
n_day_of_week
dim_pharmacies
id
name
dim_suppliers
id
name
dim_wholesalers
id
name
general_name
dim_product_names
id
name
dim_drug_groups
id
name

# Our architecture



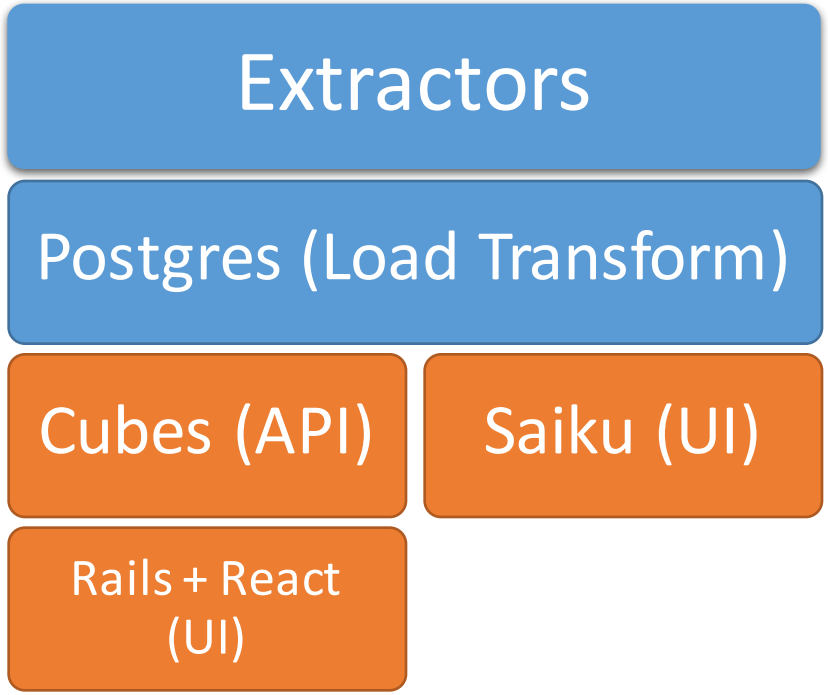
# Architecture - extractors

- Cyclone_client
  - Mssql (2008-2012)
  - Golang
  - CSV + rsync over ssh
- Kachok
  - Web scrapper
- Skytools replication
  - From existing products

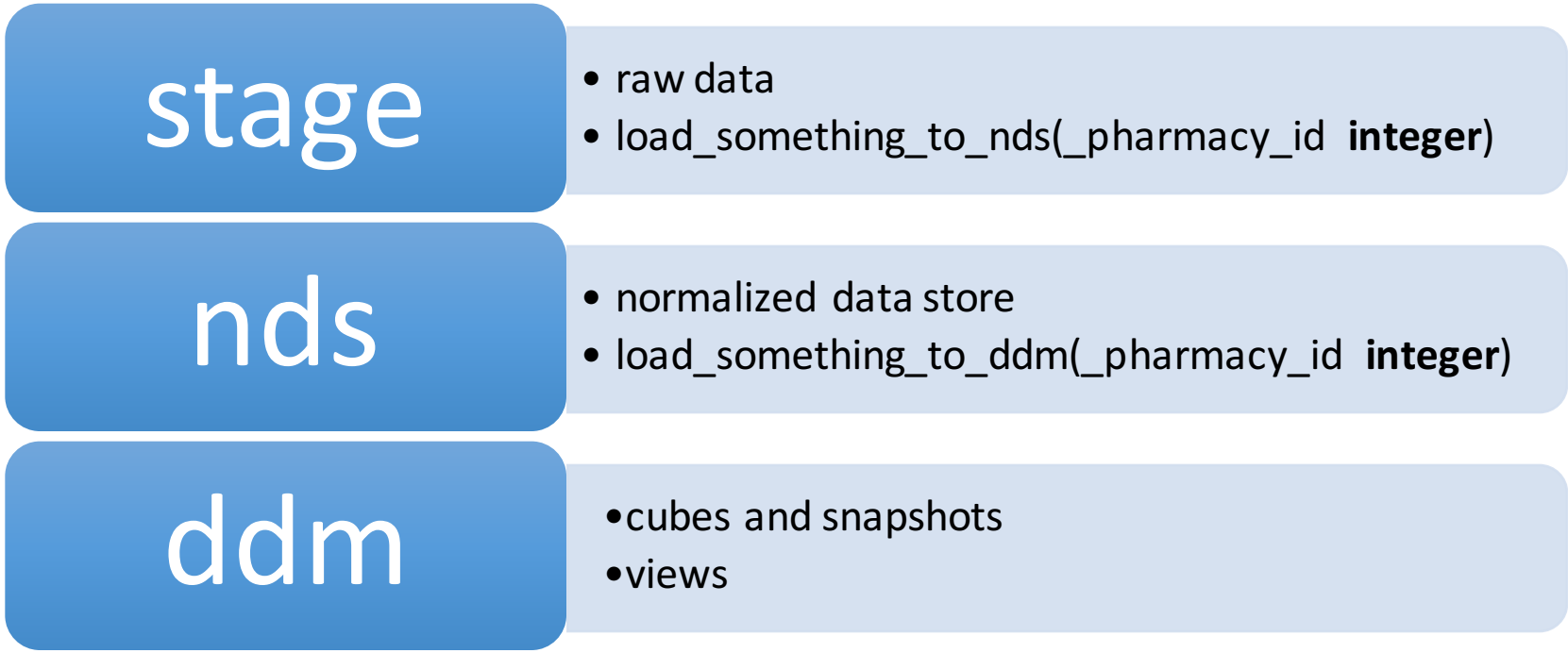


# Architecture – API + UI

- Cubes - cubes.databrewery.org
  - Easy drilling-down
  - Slicing and dicing
  - Serves aggregates, dimension details, facts
  - Provides all necessary metadata for a reporting application
- Rails, React
  - Authorization
  - d3, dc, crossfilter
- Saiku
  - Only for back office



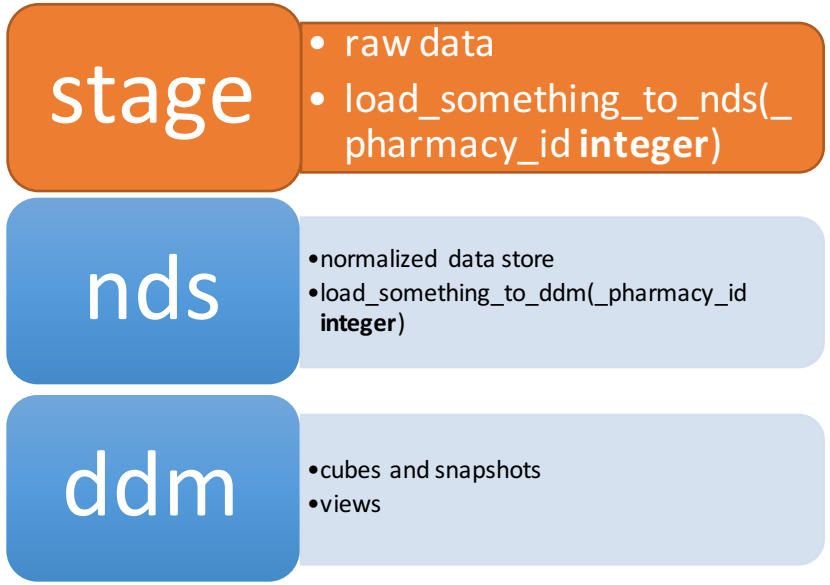
# Architecture – Postgres (load, transform)



# Architecture– Postgres (load, transform)

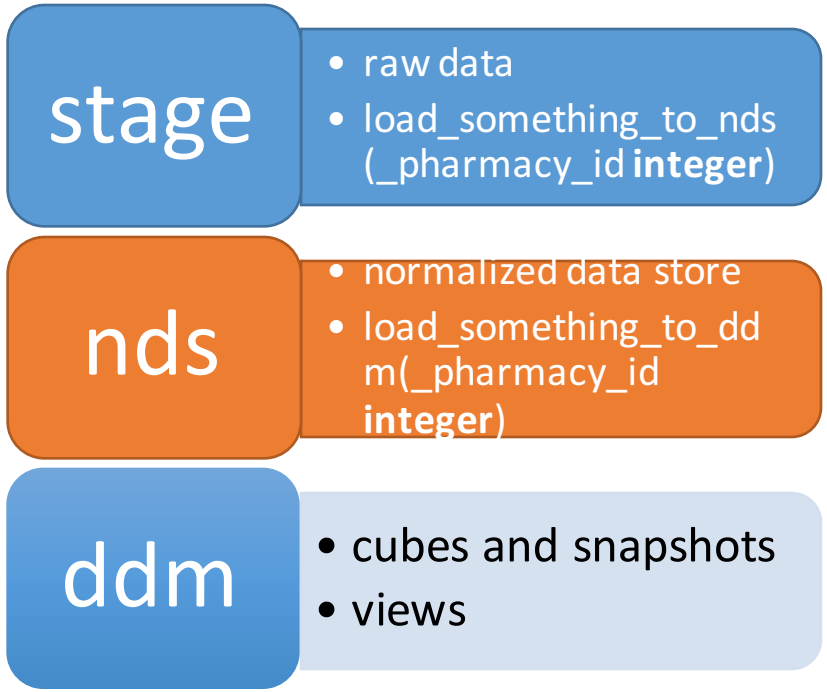
**Stage**

- «Raw» data
- Cleaned up completely in every ELT cycle
- Is as data source for NDS



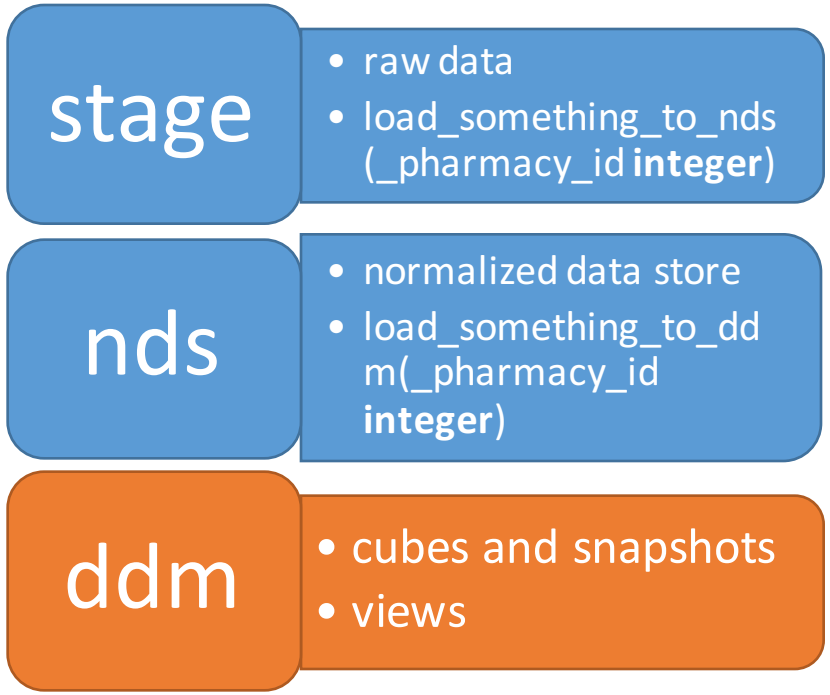
# Architecture– Postgres (load, transform)

- Normalized Data Store
  - Here data is normalizing and validating
  - Is a source for ddm
    - Measures for ddm is calculated there
  - delta calculating for loading into ddm based on last_updated field

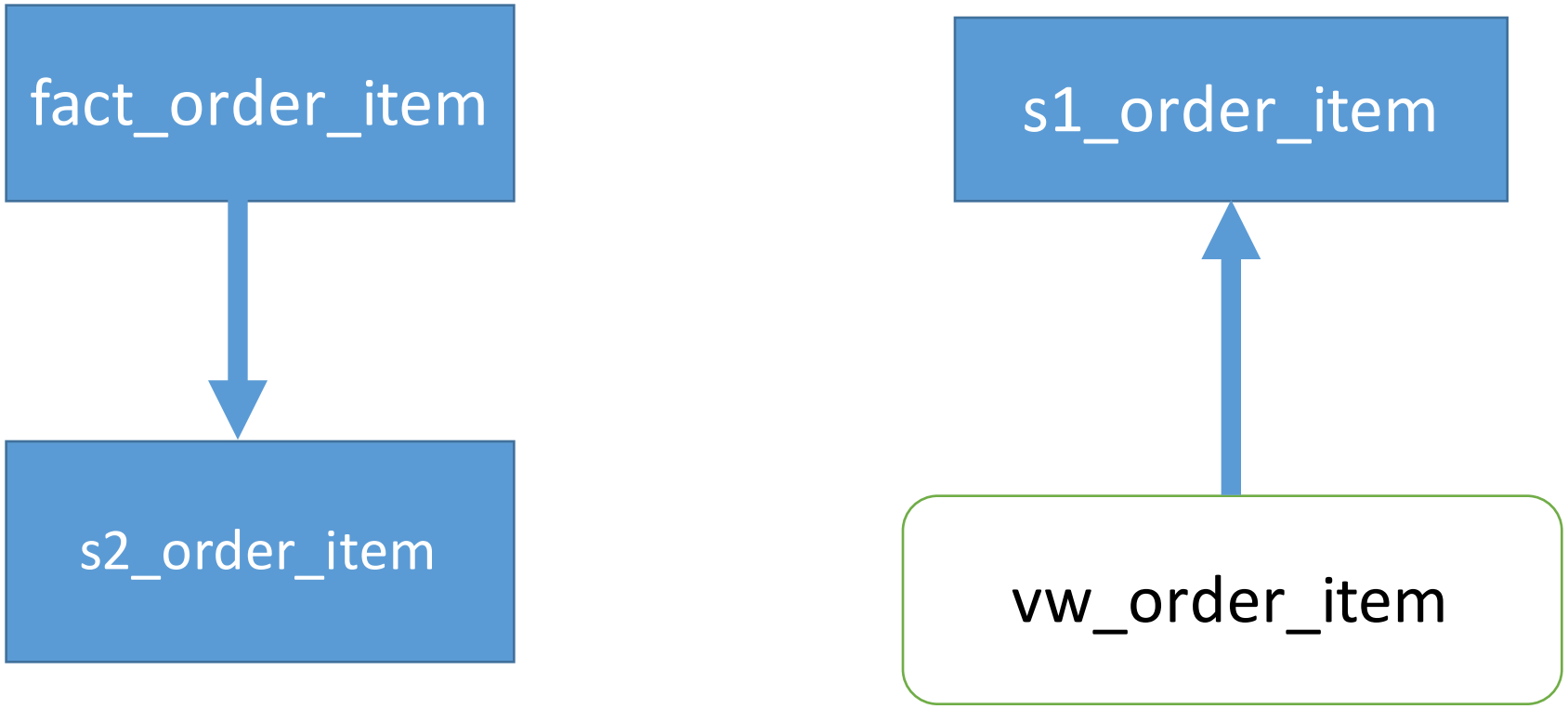


# Architecture – Postgres (load, transform)

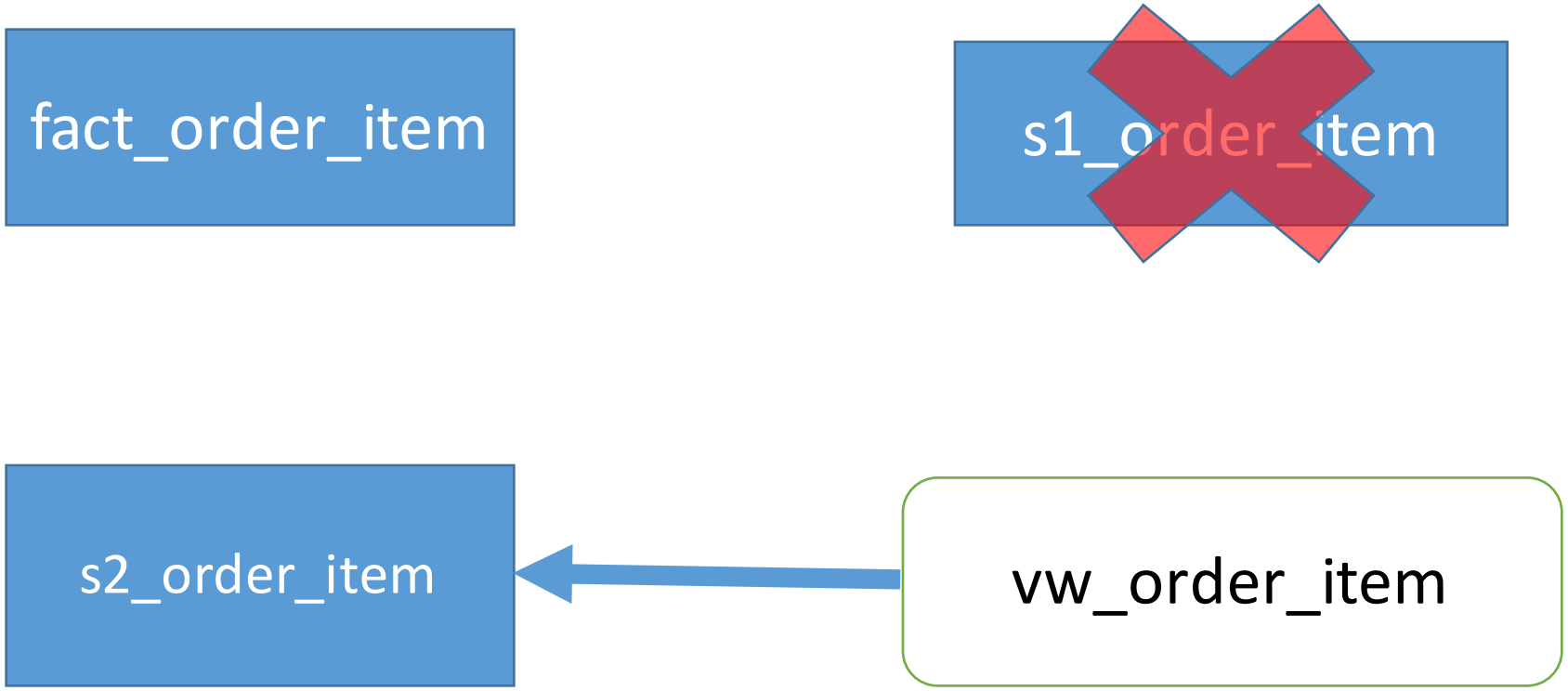
- Dimensional data model
  - Cubes
  - Snapshots
    - Deploy calmly
    - Analyze before-after release states
    - View is entry point for application



# Architecture – Postgres (snapshots)



# Architecture – Postgres (snapshots)



# Column storage

- Suitable for:
  - aggregations
  - showing fixed numbers of columns
- cstore_fdw -> https://github.com/citusdata/cstore_fdw
  - Compression: Reduces in-memory and on-disk data size by 2-4x. Can be extended to support different codecs.
  - Column projections: Only reads column data relevant to the query. Improves performance for I/O bound queries.
  - Skip indexes: Stores min/max statistics for row groups, and uses them to skip over unrelated rows.

# Column storage

- Our expirience:
  - Is not faster than vanilla postgres (say hello to cubes)
  - Volume reduced up to 12 times. Wow.
  - No way to backup traditional way (no need?)
  - No support for delete/update (snapshots)

# Configuration

- Load profile:
  - Big volume RW I/O
  - Most of  I/O is sitting in stage, nds
  - ddm is not high loaded

- shared_buffers = ½ RAM
- work_mem = 2GB
- maintenance_work_mem = 3GB
- temp_buffers = 2GB
- effective_cache_size = ½ RAM
- max_wal_sizr = 32GB

# Features

- DDM could be placed in dedicated server (londiste, pg_logical)
- Use COPY/BULK INSERTS, don’t use UPDATE (ke ke ke)
- You should think about horizontal and vertical partitioning, please find proper keys for that
- You should think about parallelism from very beginning
- Use TABLESPACES/PARTIAL INDEXES (and more and more disks)
- You should use data store policy
- Statistics should be collected in tempfs volume

# Features vol 2

- Use migrations – sqitch by theory
- You'd better test ELT - sqitch by theory
- Use pg_stat_statements (add this into monitoring)
- Use profiling – PLPROFILER3
- Sometimes, you have (not) to use cstore_fdw
- Sometimes, you have (not) to use unlogged tables

# Pros and cons

- Cons
  - No easy way to scale horizontally
  - Reasonable difficult deploy

- Pros
  - Local data (no big network transfers)
  - Effectively parallelized (thanks to pharmacy_id)
  - **PL**/pgSQL

# Thank you

andy@mastery.pro

# Speed limit

- cubes is not fast (due to serialization)
  - json (12 sec)
  - ujson (4 sec)
  - postgres json output (1.5 sec) db self time 0.3-0.7 sec