Analytical Data Modeling

(With On-Chain Data)

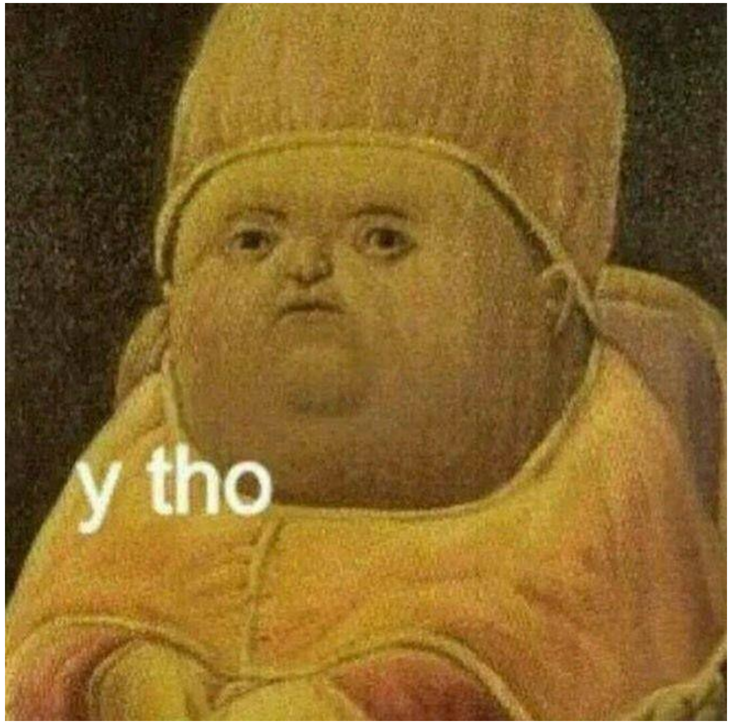
@sha2fiddy

Engineering Manager, Data Analytics

Foundry Digital

btc++ Austin 2025

Analytics Engineering



What: transform raw data into clean, reliable data models

Analogy: raw tables are ingredients; models are recipes

Why: speed of analysis, consistency, & reusability

How: applying software engineering principals and technologies to data modeling & analysis

Data Engineers: build data ingestion pipelines & infrastructure

Data Analysts: create queries, reports, & dashboards

Analytics Engineers: build data transformation pipelines with software engineering best practices, but primarily to enable analysts

Data Modeling Concepts

Data Granularity: level of detail of a data model

- *week/day/hour*
- *block/address/transaction*
- *pool/subaccount/worker*

Natural vs Surrogate Key: a unique identifier from the data source vs one generated within the data warehouse

Change Data Capture (CDC): systematically record how & when categorical data changes over time (e.g. name change)

Related: Slowly Changing Dimensions (SCDs)

Normalization vs Denormalization: separating categorical and numerical data vs combining them into flat dataset models

Star Schema Models

Star Schema: a central fact table which joins to one or more dimension tables (**normalized**)

Dimension (Dim): categorical data that is used to aggregate, filter, and sort by (one row per dim entity)

- *date, pool, customer, miner model*

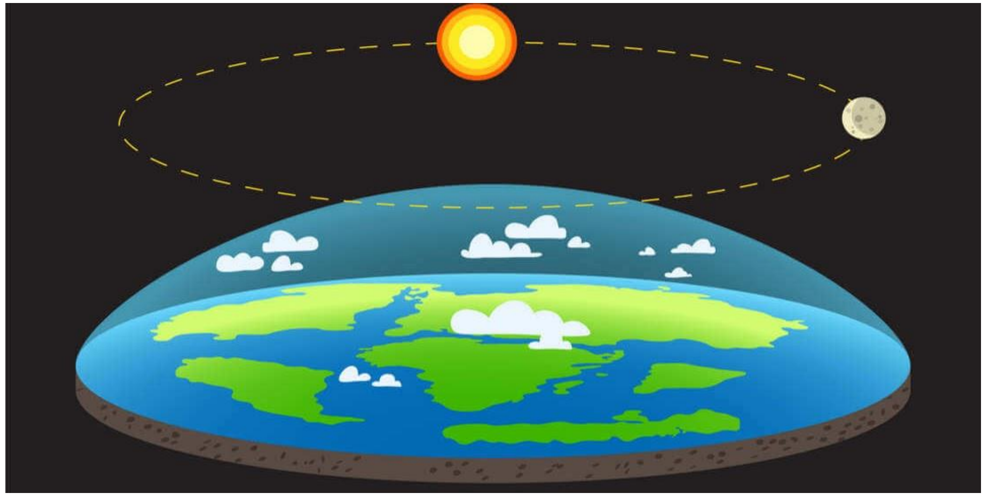
Fact: numerical measurements or events to be aggregated (one row per level of detail)

- *revenue, hashrate, price, worker count*

Pros: reduced storage, reduced code duplication, central sources of truth

Cons: joins required, less intuitive for business users

OBT Models



OBT: Operational BI Table (aka One Big Table)

Flat standalone models (**denormalized**), containing all attributes that would otherwise be broken into dims & facts

Pros: fewer joins required, more intuitive for business users

Cons: increased storage, code duplication, can introduce many categorical flags or partitions needed for filters

Best of Both: build star schemas for primary dims & facts, and application-specific OBTs which read from those dims & facts

Data Warehouse Tools & Tips

Data Build Tool (dbt): data modeling framework that brings automation, testing, & jinja templating to data transformation (core product is open source)

Analogy: like a frontend application framework, but for analytics engineering

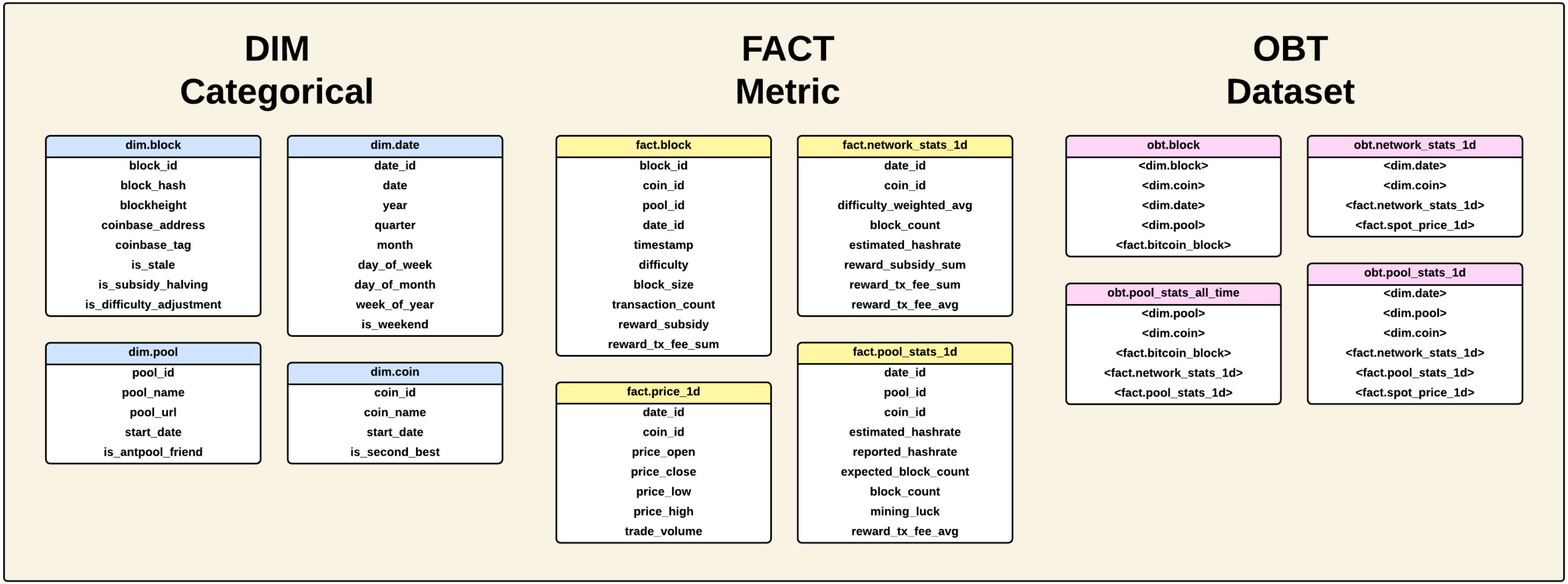
Columnar Databases: column-oriented databases are optimized for aggregating large volumes of data (vs traditional RDS systems which are transactional)

- **SaaS**: Snowflake, BigQuery, Databricks
- **Open Source**: DuckDB, ClickHouse, TimescaleDB

Extract, Load, Transform (ELT): ingest raw data to a central storage layer first, then perform all transformations (vs traditional ETL approach)

Data Validation: automated data audits & tests on raw and modeled layers help ensure data integrity & reliability

Model Examples



DIM
Categorical

dim.block
block_id
block_hash
blockheight
coinbase_address
coinbase_tag
is_stale
is_subsidy_halving
is_difficulty_adjustment

dim.date
date_id
date
year
quarter
month
day_of_week
day_of_month
week_of_year
is_weekend

dim.pool
pool_id
pool_name
pool_url
start_date
is_antpool_friend

dim.coin
coin_id
coin_name
start_date
is_second_best

FACT
Metric

fact.block
block_id
coin_id
pool_id
date_id
timestamp
difficulty
block_size
transaction_count
reward_subsidy
reward_tx_fee_sum

fact.network_stats_1d
date_id
coin_id
difficulty_weighted_avg
block_count
estimated_hashrate
reward_subsidy_sum
reward_tx_fee_sum
reward_tx_fee_avg

fact.price_1d
date_id
coin_id
price_open
price_close
price_low
price_high
trade_volume

fact.pool_stats_1d
date_id
pool_id
coin_id
estimated_hashrate
reported_hashrate
expected_block_count
block_count
mining_luck
reward_tx_fee_avg

OBT
Dataset

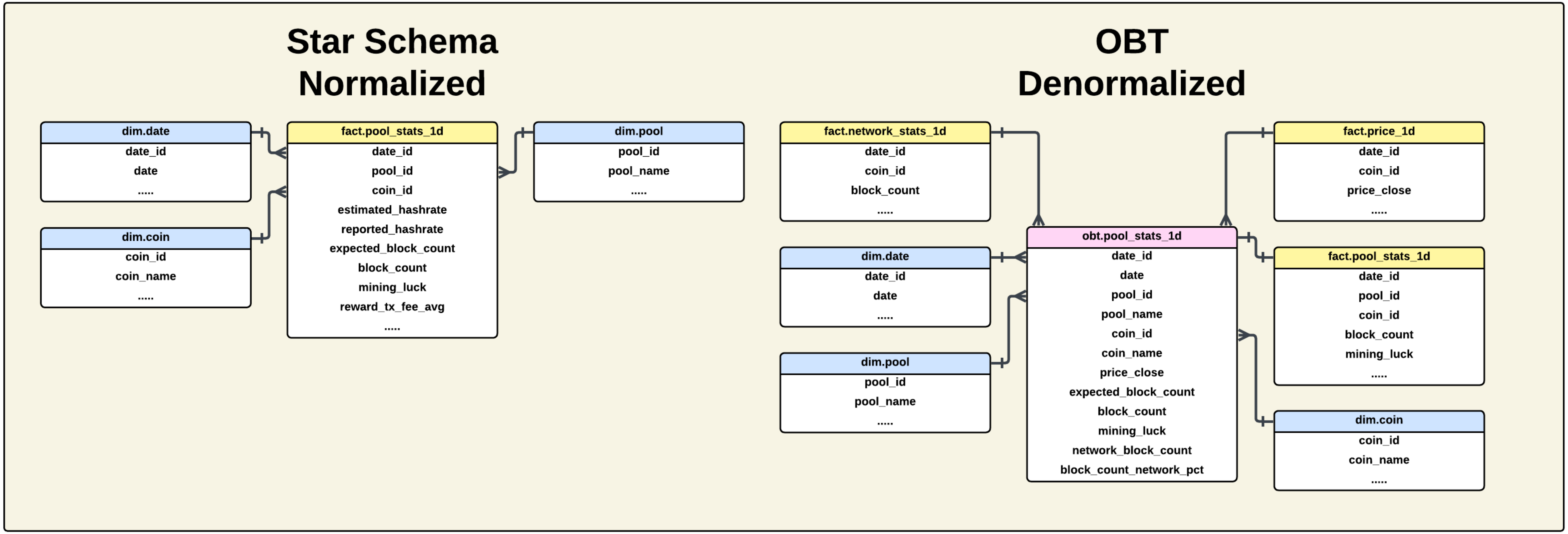
obt.block
<dim.block>
<dim.coin>
<dim.date>
<dim.pool>
<fact.bitcoin_block>

obt.network_stats_1d
<dim.date>
<dim.coin>
<fact.network_stats_1d>
<fact.spot_price_1d>

obt.pool_stats_all_time
<dim.pool>
<dim.coin>
<fact.bitcoin_block>
<fact.network_stats_1d>
<fact.pool_stats_1d>

obt.pool_stats_1d
<dim.date>
<dim.pool>
<dim.coin>
<fact.network_stats_1d>
<fact.pool_stats_1d>
<fact.spot_price_1d>

Schema Examples



Star Schema Normalized

dim.date
date_id
date
.....

dim.coin
coin_id
coin_name
.....

fact.pool_stats_1d
date_id
pool_id
coin_id
estimated_hashrate
reported_hashrate
expected_block_count
block_count
mining_luck
reward_tx_fee_avg
.....

dim.pool
pool_id
pool_name
.....

OBT Denormalized

fact.network_stats_1d
date_id
coin_id
block_count
.....

dim.date
date_id
date
.....

dim.pool
pool_id
pool_name
.....

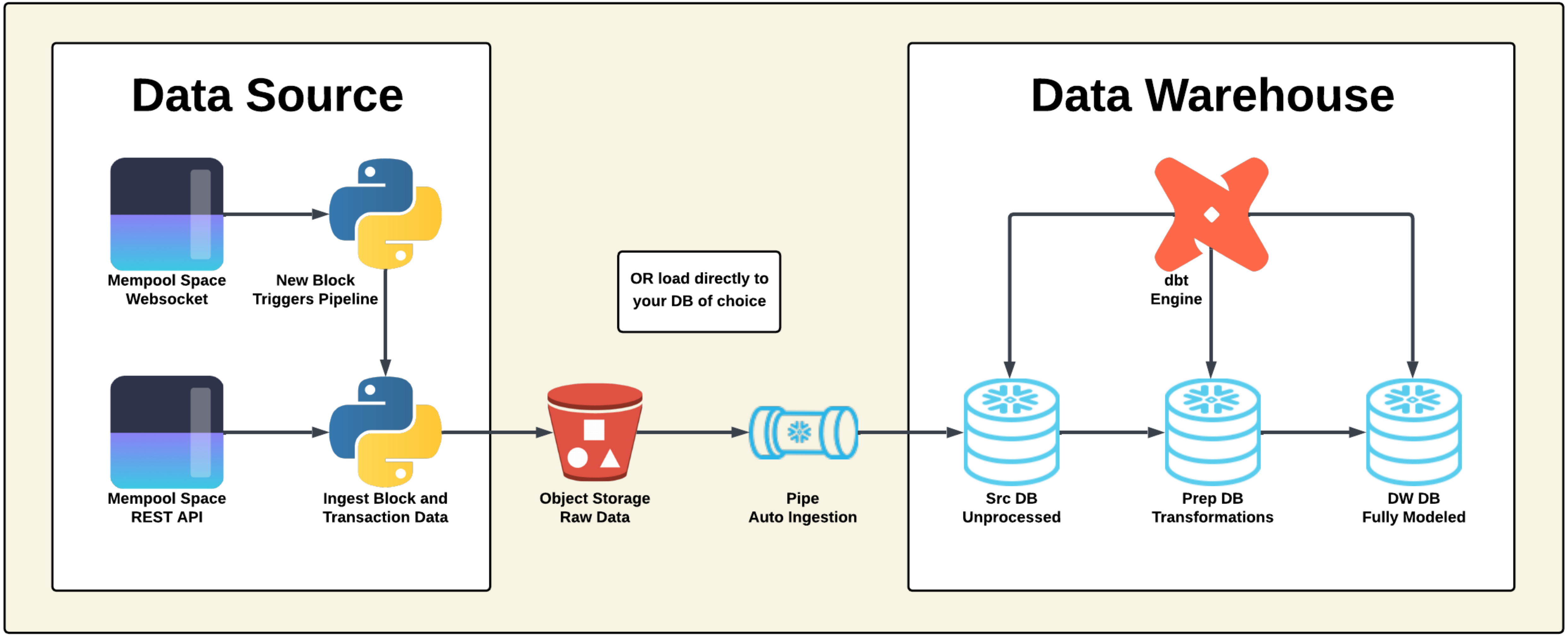
obt.pool_stats_1d
date_id
date
pool_id
pool_name
coin_id
coin_name
price_close
expected_block_count
block_count
mining_luck
network_block_count
block_count_network_pct

fact.price_1d
date_id
coin_id
price_close
.....

fact.pool_stats_1d
date_id
pool_id
coin_id
block_count
mining_luck
.....

dim.coin
coin_id
coin_name
.....

Pipeline Architecture Example



Workshop

Star Schemas:

- **Dims**: date, block, pool
- **Facts**: block, network stats, pool stats, price

OBTs: block, network stats, pool stats

Metrics: estimated network hashrate, hashprice, pool mining luck

Repo: https://github.com/sha2fiddy/btcpp-demo-2025