INTRO TO DATA WAREHOUSES

"DW is a copy of transactions data specifically structured for query and analysis"

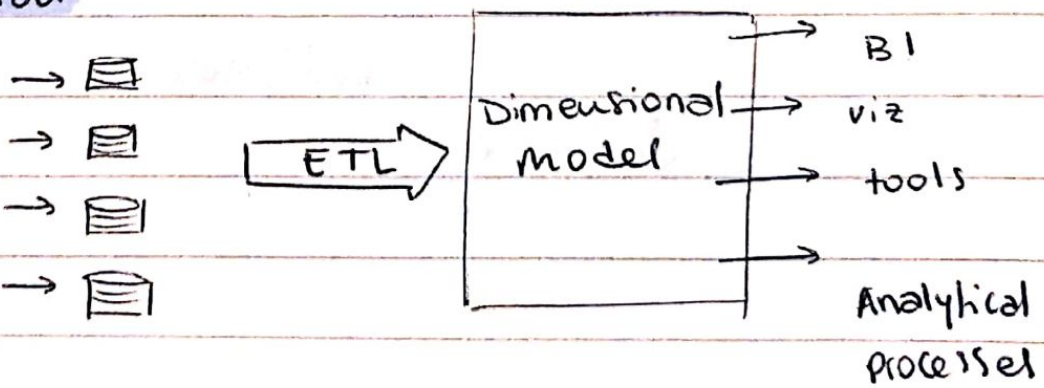
- Kimball

"Dw is a subject-oriented, integrated, non-volatile, time-variant collection of data in support of management's decisions"

- Inmon

"Dw is a system that retrieves and consolidates data periodically from source systems into dimensional or normalized data store. It keeps years of history and is queried for business intelligence. Typically updated in batches"

- Rainardi



ETL

pip install ipython-sql

%load_ext sql ← load

%sql <one lines query> $variables

%%sql <multiline query>

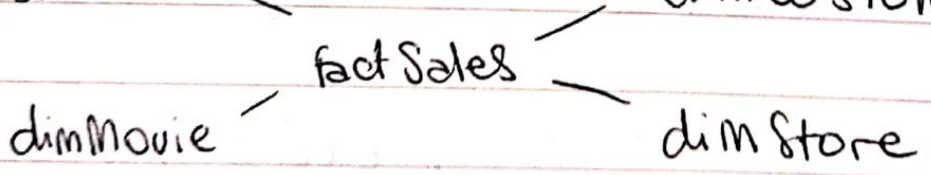
$conn-string → = "postgresql://..."

%sql $conn-string

1°) Defino la estructura del modelo dimensional

La idea es extraer data de relational DB en 3NF donde tengo que hacer muchos JOINS para obtener la info que quiero a

→ Dim - Fact Tables. por ejemplo:



2º) creo las tablas dimensiones y hechos

Para crear las tablas:

%% SQL

```
CREATE TABLE dimDate
{   date_key SERIAL BRIMARY KEY,
    date date NOT NULL,
    year integer NOT NULL, ...
}
```

%%% SQL

```
CREATE TABLE FactSales
{   sale_key SERIAL PRIMARY KEY,
    date_key integer REFERENCES
        dimDate (date_key),
    ...,
    sales_amount numeric NOTNULL
}
```

Referencias a dimensiones

Facts / metricas

3º) ETL data de tablas 3NF a Facts y Dims

%%% SQL

(L) ← INSERT INTO dimDate (date_key, date,
year, ..., is_weekend)

(E) ← SELECT DISTINCT (TO_CHAR (payment_date::DATE
'YYYYMMDD')::integer) AS date_key,
EXTRACT (year FROM payment_date)
AS year,
(T) date (payment_date) AS date,
...
CASE WHEN EXTRACT (ISODOW FROM
payment_date) IN (6,7) THEN TRUE
ELSE FALSE
FROM payment;

we denormalize tables (put joins on our dim tables) and transform the data.

D.W. ARCHITECTURE

- Kimball's Bus Architecture
- Independent Data Marts
- Inmon's Corporate Information Factory
- Hybrid Bus + CIF

KIMBALL'S BUS ARCHITECTURE

source transactions

← BACK ROOM → "kitchen"

ETL Systems:
- Transform from source to target
- conform dimensions
- NO user query support

← FRONT ROOM → "dining room"

Presentation Area
- Dimensional
- Atom + Sumary data
- Organized by (business process)
- uses conformed dimensions

Applications
- Ad hoc Queries
- standard reports
- Analytics apps

Design Goals:
Ease of use
Query performance

NOT departments- ex.
→ invoice processing
→ register
→ product etc.

BUS MATRIX

Business processes x Dimensions table

	Date	Product	Warehouse	...
BP1	x		x	
BP2	x	x		
:				
			x	

- Dimensions are across business departments

ETL

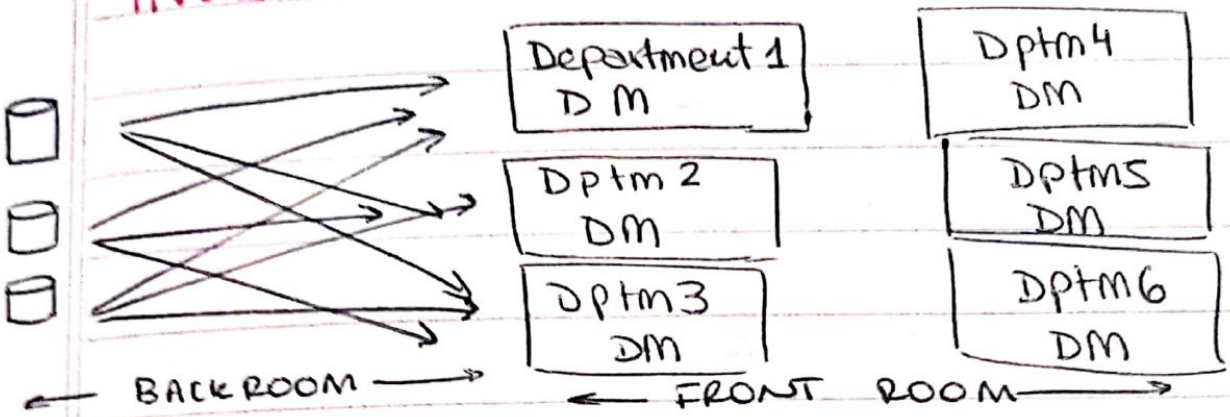
→ EXTRACT · Get data from <> sources , OLTP
· Delete stale/old data from sources

→ TRANSFORMATION · Integrate <> sources together
· Data cleansing : inconsistencies, duplications, missing values, etc.
· Produce diagnostic metadata (this source has % of unreliable data)

→ LOADING · Structure and load .. data to dimensional data model.

→ 1 Dimensional data model shared by all the departments, organized by business process.

INDEPENDENT DATA MARTS



Departments have independent ETL processes and dimensional tables

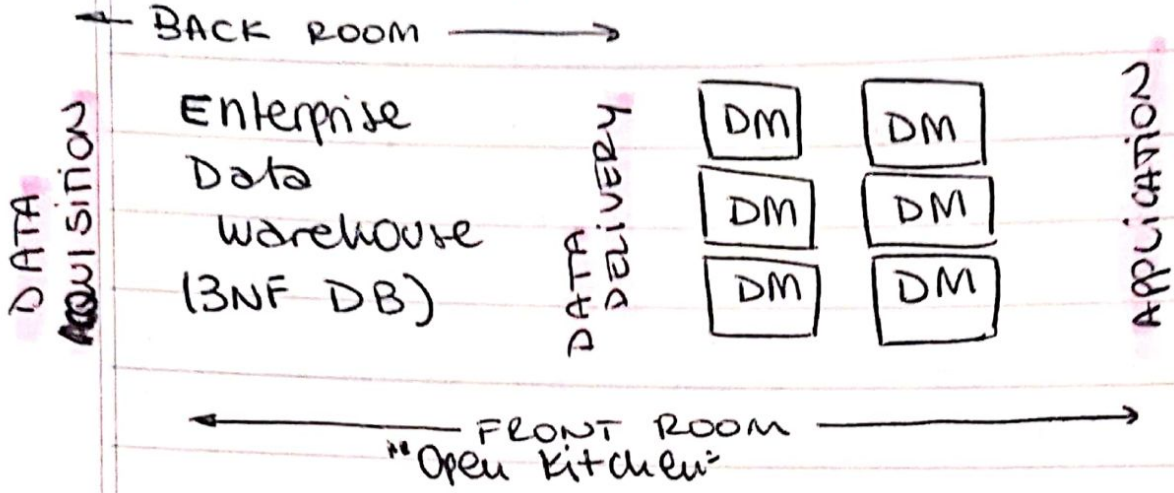
These separated dim tables are called DM.

Each have its own fact tables for the same event → no conformed dimensions.

Uncoordinated efforts might lead to inconsistent views

It happens ad-hoc because it's simpler and autonomous, but it's highly discouraged.

INMON'S CORP. INFORMATION FACTORY (CIF)

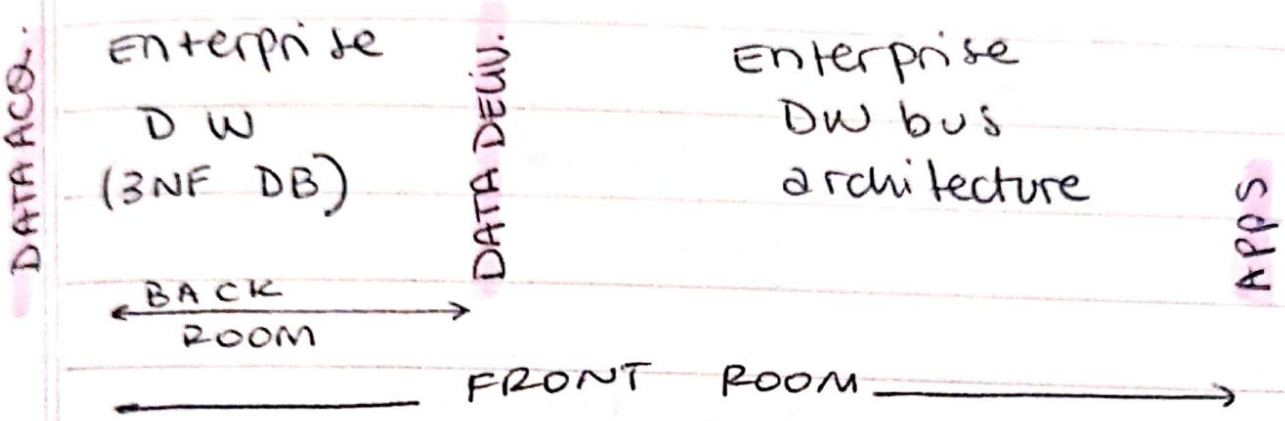


2 ETL processes → 1°: Source → DW (3NF DB)
2°: DW → DM's

3NF DB acts as an enterprise wide data store
- → Single integrated source of truth for users
- → Accessible by end users

DM's are dimensionally modelled, and contain mostly aggregated data.

HYBRID KIMBALL BUS & INMON CIF



↓

Conformed dimensions by business processes.

A bit of both worlds

When you feel like quitting ...
Think about why you started.

OLAP CUBES

Aggregation of fact metric on a number of dimensions.

Eg. Movie, Branch, Month

	APR			
	MAR	NY	Paris	SF

FEB	NY	Paris	SF
Avatar	$25k	$5k	$15k
starwars	$15k	$7k	$10k
Batman	$3k	$2k	$3k

- Easy to communicate to end users

Roll-up: Sums the sales of each city by country (less columns in branch dim)

eg.

FEB	US	FR	ARG	...
Avatar				
starwars				
Batman				

Groups up a dimension, and sums the facts

Drill-down: Decompose sales of each city into smaller districts (more columns in branch dimension).

⇒ OLAP cubes should store the finest grain of data (atomic data) in case we need to drill-down to the lowest level.

eg. Country → City → District → Street.

Slice: Reduce N dimensions to N-1 dimensions by restricting one dimension to a single value.

Eg. month = "MAR"

→ I will be seeing only a 2dim table.

Dice: Same dimensions but computing a sub-cube by restricting some values of the dimensions

Eg. month in ['FEB', 'MAR'] and
movie in ['Batman', 'Avatar'] and
branch ['NY']

→ 'smaller sub cubes'

OLAP CUBES Query Optimization

Business users will typically want to slice, dice, rollup, drill down all the time
→ Ad hoc Queries

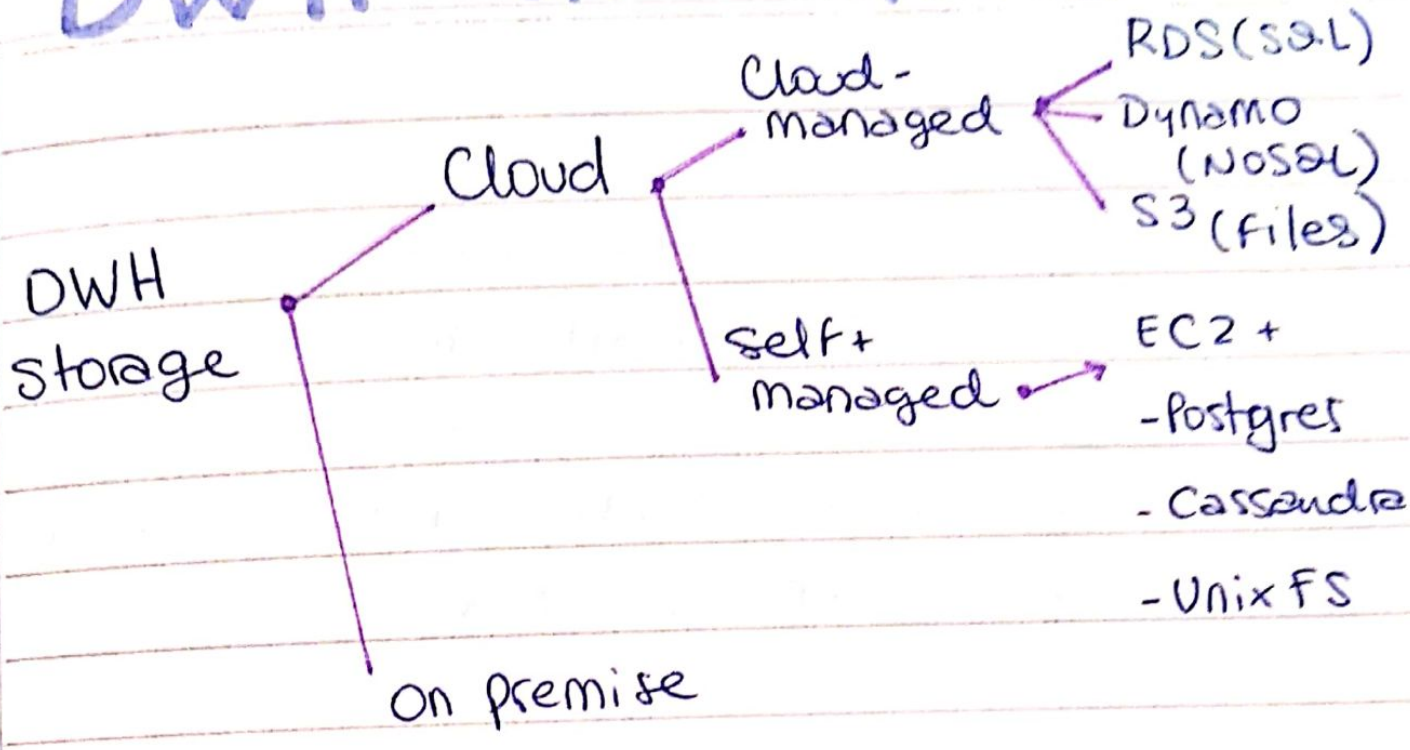
Each combination/operation will potentially go through all the fact table
→ Sub optimal

* GROUP BY CUBE (movie, branch, month)* will make (1) pass through the facts table and aggregate all possible combs. of groupings, of length 0, 1, 2, 3, etc.

∅	1	2	3
• Total revenue	• Revenue by - movie - branch - month	• Revenue by - movie, branch - branch, month - movie, month	• Revenue by movie, branch, month

Saving/Materializing the output of the CUBE operation and using it → enough to answer all forthcomming agg. from business users w/o processing the whole Facts table.

DWH ON AWS



AMAZON REDSHIFT

MPP: massively parallel processing DB.

→ parallelizes query execution on multiple CPU / machines

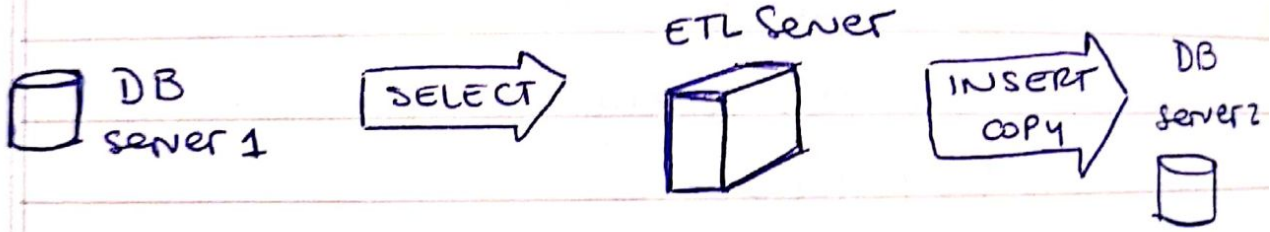
Tables are partitioned, and each partition is processed in parallel

- Column oriented storage.
- It's a modified Postgresql.

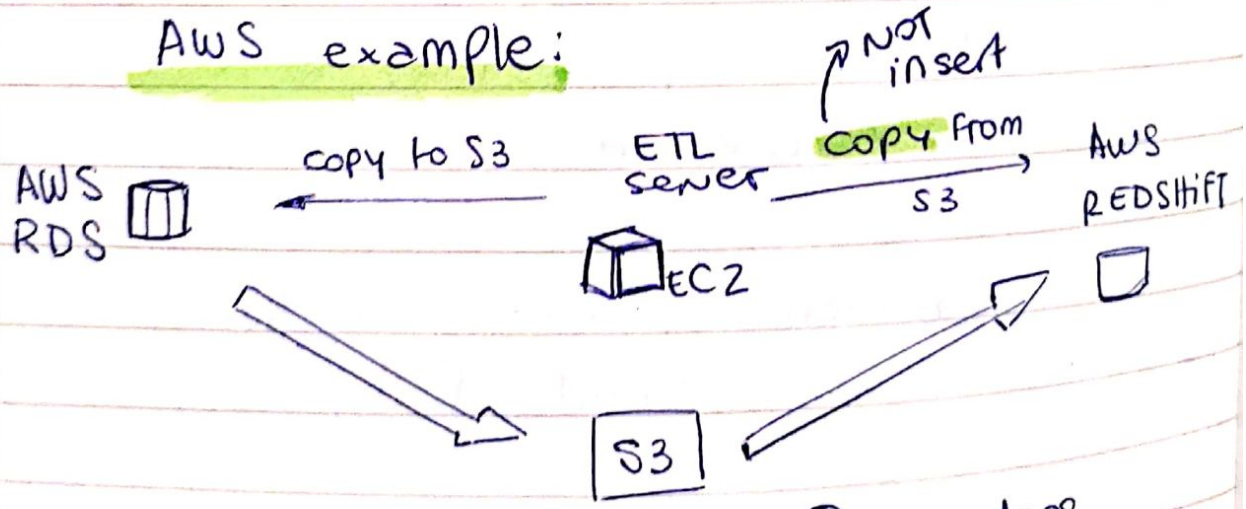
SQL TO SQL

```
Select fact1, fact2
INTO   newFact Table
FROM   table X, Y
WHERE  X.id = Y.fid, X, Y <> NULL
GROUP BY  Y.d
```

IF (because vendors) we cannot insert diredly into new table (if we are using a diffent sever);



AWS example:



* copy does bulk insertion → beter 4 scale

```
COPY sporting_event_ticket FROM
's3://udacity-labs/tickets/split/part'
CREDENTIALS '...'
gzip DELIMETER ';' REGION 'us-west-2';
```

va a copiar todas las particiones que empiezan con ese key.
Para ser más concreta:

cust. manifest

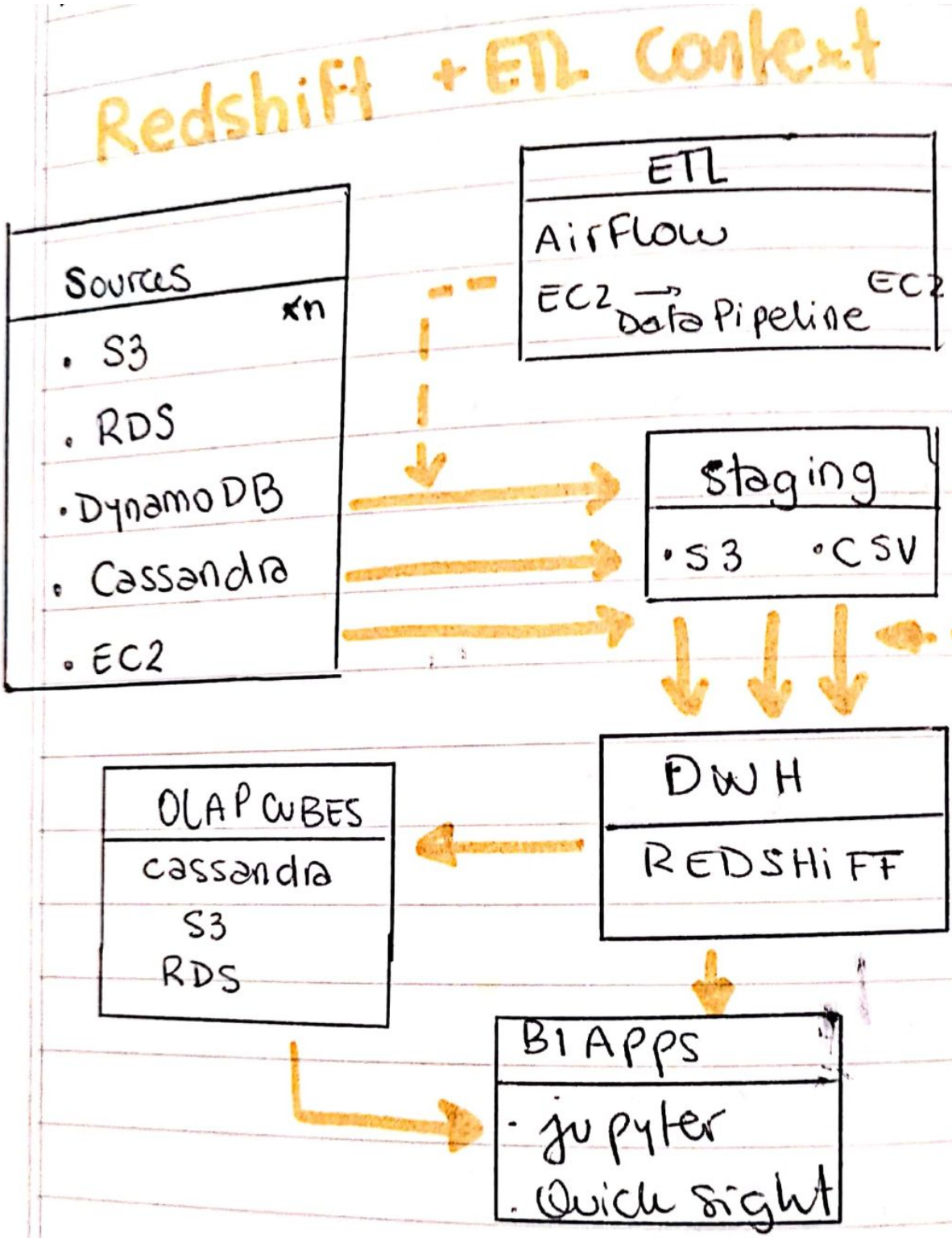
```
{ "entries": [
    {"url":"..." , "mandatory": true },
    ... ] }
```

```
COPY customer
FROM 's3://mybucket/cust.manifest'
IAM_ROLE '...'
manifest;
```

ETL out

```
UNLOAD ('select * from venue limit 20')
to 's3://mybucket/venue_pipe'
iam_role '...';
```

Redshift + ETL context



Infrastructure as a Code (IaC)

- aws-cli scripts
- AWS sdk (ex. Python → boto3)
- Amazon Cloud formation

Benefits.

→ sharing → Multiple deployments
→ Reproducibility → Maintainability

1° Create user w/ admin privileges
2° with boto3 + credentials create clients for IAM, EC2, S3, Redshift.

```
ec2 = boto3.resource('ec2', region, creds)
```

3° Get bucket: S3.Bucket("name")
4° Create IAM Role to allow Redshift read access to S3
5° Create Redshift Cluster w/ created Role
(+) Params: clustey Type, Node Type, Nº Nodes, DBName, cluster ID, Master Username + Pass.

6° Open incomind TCP port to cluster

Get Redshift cluster's VPC id, endpoint, etc.

+ Get VPC of cluster's vpc id

→ Get security groups of VPC

→ Auth. ingress to SG

TOandFrom DWH - Port

7° Connect to DB

% SQL 'postgresql : // DBUser : Pass @Endpoint : Port/ DB'

8. CREATE Table.

9. LOAD data partitioned to cluster

COPY DB-NAME FROM 'S3 : // bucket/parts'

Credentials aws-iam-role =

gzip delimeter ';'

region ;

Table Design Optimization

Based on Partition strategies based on frequent access pattern.

→ DISTRIBUTION STYLE

- EVEN: Table is partitioned on slices such that each slice would have almost equal no. of records from partitioned table
- ALL or BROADCASTING: Replicates a table on all slices
- AUTO: Allow Redshift to determine distrib. style.
- KEY: Rows having similar values are placed in the same slice.

→ SORTING KEY

Define a column as sort key.