DW #2 — Fabrication (OEE) : de RAW → staging → dimensions → faits → OEE + Extensions

Programme Data Engineer

ECE

23 octobre 2025

Objectifs du Mini-TP et plan

Objectifs :

- Simuler des données **RAW** de fabrication : machines, produits, shifts, production, arrêts (downtime).
- Nettoyer/normaliser (*staging*) et construire les **dimensions** (date, machine, produit, shift).
- Construire deux tables de **faits** : `fact_production` (additive) et `fact_downtime` (semi-additive).
- Calculer l'**OEE** (Availability, Performance, Quality) par mois et tracer le graphique.

Plan : Contexte →RAW →Staging →DIM →FACT →Agrégations/OEE →Graphique →TP →**Extensions (SCD2, Accumulating Snapshot, Bridge, LAF, DQ)**.

Contexte et setup (imports, reproductibilité)

```python
import numpy as np, pandas as pd
from datetime import datetime, timedelta
from pathlib import Path
import matplotlib.pyplot as plt

rng = np.random.default_rng(7)
out_dir = Path("dw_mfg"); out_dir.mkdir(exist_ok=True)

start, end = datetime(2025,1,1), datetime(2025,9,30)
dates = pd.date_range(start, end, freq="D")
```

Génération RAW : référentiels (machines, produits, shifts)

```
areas = ["Cutting","Assembly","Packaging"]
raw_machines = pd.DataFrame({
  "machine_id": range(1, 16),
  "machine_name": [f"M{i:02d}" for i in range(1,16)],
  "area": rng.choice(areas, 15),
  "install_date": pd.to_datetime("2022-01-01") + pd.to_timedelta(rng
      .integers(0,365,15), unit="D")
})

raw_products = pd.DataFrame({
  "sku": [f"SKU{i:03d}" for i in range(1,41)],
  "product_name": [f"Prod {i:03d}" for i in range(1,41)],
  "ideal_rate_per_min": rng.choice([0.8,1.0,1.2,1.5], 40)  # unit s
      /min
})

raw_shifts = pd.DataFrame({
  "shift_id": [1,2,3],
```

Génération RAW : production et arrêts (downtime)

```
# Production journali re par machine + shift + produit
prod_rows = []
for d in dates:
  for m in raw_machines["machine_id"]:
    for s in raw_shifts["shift_id"]:
      if rng.random() < 0.85:  # shift r ellement produit ?
        sku = rng.choice(raw_products["sku"])
        good = rng.integers(200, 800)
        scrap = rng.integers(0, int(good*0.15))
        prod_rows.append([d, m, s, sku, good, scrap])
raw_production = pd.DataFrame(prod_rows, columns=[
  "prod_date","machine_id","shift_id","sku","units_good","
    units_scrap"
])

# Arr ts (downtime) journaliers
failure_modes = ["Setup","Breakdown","Quality","Material","
   PlannedMaintenance"]
```

Staging : nettoyage standardisation

```
stg_machines = raw_machines.assign(
  machine_name = raw_machines["machine_name"].str.upper(),
  area = raw_machines["area"].str.title()
)

stg_products = raw_products.assign(
  product_name = raw_products["product_name"].str.title()
)

stg_shifts = raw_shifts.copy()

stg_production = raw_production.assign(
  prod_date = pd.to_datetime(raw_production["prod_date"]),
  units_total = lambda df: df["units_good"] + df["units_scrap"]
)

stg_downtime = raw_downtime.assign(
  down_date = pd.to_datetime(raw_downtime["down_date"]),
```

Dimensions : dim_date, dim_machine, dim_product, dim_shift

```
# dim_date : calendrier continu
dim_date = pd.DataFrame({"date": pd.date_range(stg_production["
    prod_date"].min(),
                                                   stg_production["
                                                      prod_date"].max(),
                                                       freq="D")})
dim_date["date_sk"] = np.arange(1, len(dim_date)+1)
dim_date = dim_date.assign(
  day=dim_date["date"].dt.day,
  month=dim_date["date"].dt.month,
  quarter=dim_date["date"].dt.quarter,
  year=dim_date["date"].dt.year,
  dow=dim_date["date"].dt.dayofweek
)

# dim_machine
```

Tables de faits : fact_production et fact_downtime

```
# Keys de date
prod = stg_production.merge(dim_date[["date","date_sk"]],
                            left_on="prod_date", right_on="date").
                               drop(columns=["date"])
prod = (prod
 .merge(dim_machine[["machine_id","machine_sk"]], on="machine_id")
 .merge(dim_product[["sku","product_sk","ideal_rate_per_min"]], on="
    sku")
 .merge(dim_shift[["shift_id","shift_sk","planned_minutes"]], on="
    shift_id"))

# Fait production (additive)
fact_production = prod[[
 "date_sk","machine_sk","shift_sk","product_sk",
 "units_good","units_scrap","units_total","ideal_rate_per_min","
    planned_minutes"
]]
```

Agrégations : Availability, Performance, Quality, OEE

```
# Agrégations par jour (toutes machines)
daily_prod = (fact_production
  .merge(dim_date[["date_sk","year","month","date"]], on="date_sk")
  .groupby(["date","year","month"], as_index=False)
  .agg(units_good=("units_good","sum"),
       units_total=("units_total","sum"),
       planned_minutes=("planned_minutes","sum"),
       ideal_rate=("ideal_rate_per_min","mean")))  # simplification

daily_down = (fact_downtime
  .merge(dim_date[["date_sk","date"]], on="date_sk")
  .groupby("date", as_index=False)["minutes_down"].sum())

daily = daily_prod.merge(daily_down, on="date", how="left").fillna({
    "minutes_down":0})

# Indicateurs
daily["availability"] = (daily["planned_minutes"] - daily["
```

Graphique : OEE mensuel (matplotlib)

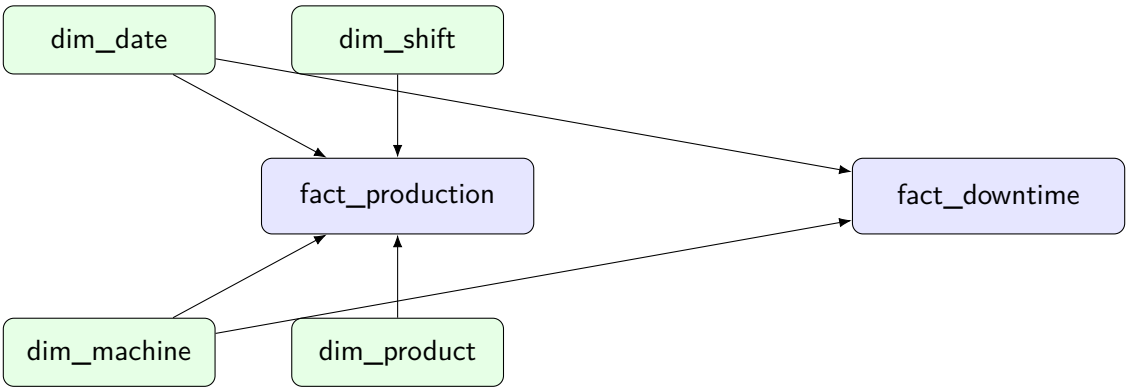
```
x = range(len(oee_month))
labels = [f"{int(y)}-{int(m):02d}" for y,m in zip(oee_month["year"],
    oee_month["month"])]
plt.figure(); plt.plot(x, oee_month["oee"], marker="o")
plt.title("OEE mensuel (exemple)")
plt.xlabel("Mois"); plt.ylabel("OEE")
plt.ylim(0,1); plt.xticks(x, labels, rotation=45, ha="right")
plt.tight_layout(); plt.savefig(out_dir/"monthly_oee.png"); plt.show
    ()

# Persist
dim_date.to_csv(out_dir/"dim_date.csv", index=False)
dim_machine.to_csv(out_dir/"dim_machine.csv", index=False)
dim_product.to_csv(out_dir/"dim_product.csv", index=False)
dim_shift.to_csv(out_dir/"dim_shift.csv", index=False)
fact_production.to_csv(out_dir/"fact_production.csv", index=False)
fact_downtime.to_csv(out_dir/"fact_downtime.csv", index=False)
```

TP : exercices proposés

- Calculer l'OEE par **machine** et par **zone** (area). Identifier les 5 machines les moins performantes.
- Décomposer l'OEE en **A/P/Q** par mois et tracer 3 courbes.
- Top 10 **modes de défaillance** par minutes perdues ; Pareto 80/20.
- Construire une **vue matérialisée** `v_oee_monthly(area)`.
- Ajouter des **règles DQ** : `minutes_down >= 0`, `units_total >= units_good`, `ideal_rate_per_min > 0`.

Schéma en étoile — vue Production/Downtime



Extension 1 — SCD2 sur dim_machine

Objectif : historiser les changements (ex. area, status) d'une machine.

```sql
CREATE TABLE dim_machine (
  machine_sk        BIGINT PRIMARY KEY,
  machine_id        INT NOT NULL,
  machine_name      VARCHAR(40) NOT NULL,
  area              VARCHAR(40) NOT NULL,
  status            VARCHAR(20) DEFAULT 'Active',
  valid_from        DATE NOT NULL,
  valid_to          DATE NOT NULL,
  is_current        BOOLEAN NOT NULL,
  CONSTRAINT uk_machine_hist UNIQUE(machine_id, valid_from)
);

-- Règle : pour un changement, fermer l'ancienne ligne (valid_to = new_from - 1, is_current=false)
-- et insérer la nouvelle (valid_from = date_changement, valid_to = '9999-12-31', is_current=true).
-- Les faits pointent sur machine_sk via une table de mapping date
```

Extension 2 — Accumulating Snapshot : fact_workorder

Objectif : suivre le cycle de vie d'un ordre de maintenance (ouverture →clôture).

```
CREATE TABLE fact_workorder (
  wo_sk             BIGINT PRIMARY KEY,
  wo_number         VARCHAR(30) NOT NULL,    -- d g n r e
  machine_sk        BIGINT NOT NULL,
  open_date_sk      BIGINT,
  schedule_date_sk  BIGINT,
  start_date_sk     BIGINT,
  complete_date_sk  BIGINT,
  close_date_sk     BIGINT,
  labor_hours       NUMERIC(10,2),
  material_cost     NUMERIC(12,2),
  status            VARCHAR(20)
);
-- Colonnes de dates remplies au fil des vnements ; mesures mises
      jour (upsert).
```

Extension 3 — Bridge table : multi-causality des arrêts

Objectif : un arrêt peut avoir plusieurs causes. Utiliser un **bridge** avec pondération.

```sql
CREATE TABLE dim_failure_mode (
  failure_mode_sk BIGINT PRIMARY KEY,
  failure_mode    VARCHAR(60) UNIQUE
);

CREATE TABLE br_downtime_cause (   -- bridge
  downtime_id     BIGINT NOT NULL,  -- identifiant du fait downtime (ou surrogate propre)
  failure_mode_sk BIGINT NOT NULL,
  allocation_pct  NUMERIC(5,4) NOT NULL CHECK (allocation_pct >= 0 AND allocation_pct <= 1),
  PRIMARY KEY (downtime_id, failure_mode_sk)
);
-- Somme des allocation_pct = 1 par downtime_id.
-- Les agr gations de minutes se font en r partissant minutes_down * allocation_pct.
```

Extension 4 — Late Arriving Facts & Unknowns

- Créer des **membres inconnus** (SK = 0 ou -1) dans chaque dimension avec attributs "Unknown".
- Charger les faits en pointant vers SK=0 si la dimension n'est pas encore disponible.
- **Rattrapage** : dès que la dimension arrive, *fixer* le SK dans le fait (update sécurisé).

```
1 -- Exemple membre inconnu
2 INSERT INTO dim_product(product_sk, sku, product_name,
     ideal_rate_per_min)
3 VALUES (0, 'UNK', 'Unknown Product', 1.0) ON CONFLICT DO NOTHING;
```

Extension 5 — Data Quality : règles rapport d'anomalies

```sql
-- Exemples de r gles
ALTER TABLE fact_downtime
  ADD CONSTRAINT ck_minutes_nonneg CHECK (minutes_down >= 0);

-- Rapport d'anomalies (extrait)
CREATE VIEW dq_anomalies AS
SELECT 'NEGATIVE_MINUTES' AS rule, COUNT(*) AS n
FROM fact_downtime WHERE minutes_down < 0
UNION ALL
SELECT 'SCRAP_GT_TOTAL', COUNT(*)
FROM fact_production WHERE units_scrap > units_total
UNION ALL
SELECT 'ZERO_OR_NEG_RATE', COUNT(*)
FROM dim_product WHERE ideal_rate_per_min IS NULL OR
   ideal_rate_per_min <= 0;
```