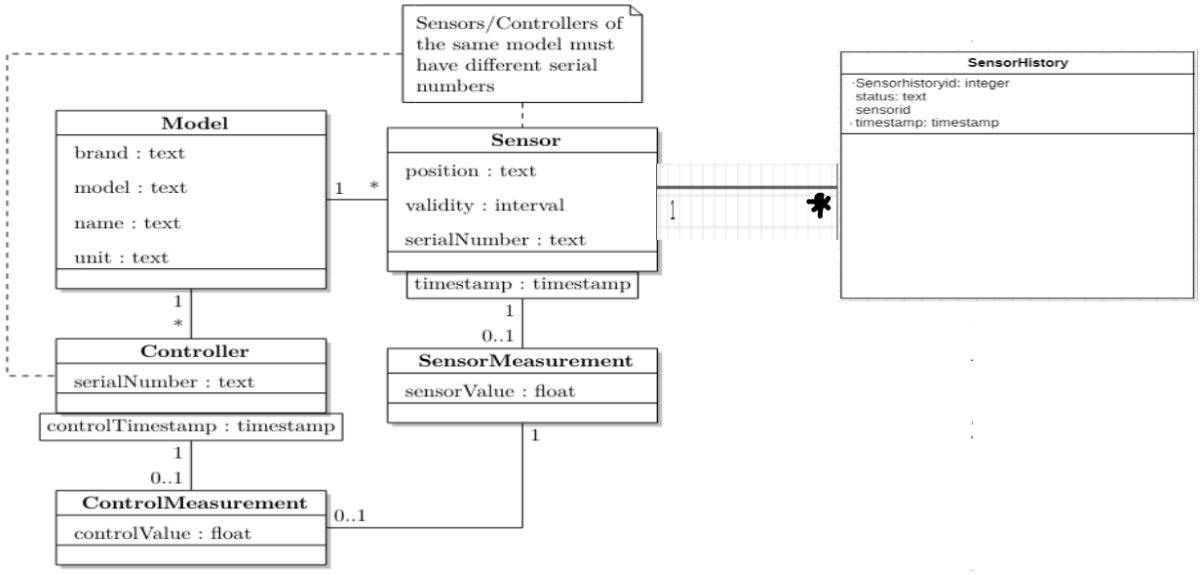
SAE 2.04 : Carte Du Controle

Projet: Section 5 du projet

Réalisé par TRAN Minh-Tue
Membre:
SAINZ Miquel
SIDHOUM Sid-ali

Base de données

Diagramme de classe extension du la figure 3:



**Notation : type de Sensor Storyid est Serial(integer va être créé automatique)

Voici c’est comment j’ai créer notre propre BDD:

La création pour la table SensorHistory:

```
CREATE TABLE SensorHistory (
   sensorHistoryId SERIAL PRIMARY KEY,
   sensorid INT REFERENCES Sensor(sensorid),
   status TEXT CHECK (status IN ('in service', 'recalibrated',
'decommissioned')),
   timestamp TIMESTAMP
);
```

L’insertion pour la table SensorHistory.

```
INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES
    (1, 'in service', '2022-01-01 10:00:00'),
    (1, 'recalibrated', '2022-06-01 10:00:00'),
    (1, 'recalibrated', '2023-01-01 10:00:00'),
    (1, 'decommissioned', '2024-01-01 10:00:00'),
    (2, 'in service', '2023-01-01 10:00:00'),
    (2, 'recalibrated', '2023-07-01 10:00:00'),
    (2, 'recalibrated', '2024-01-01 10:00:00');
```

Les requetes

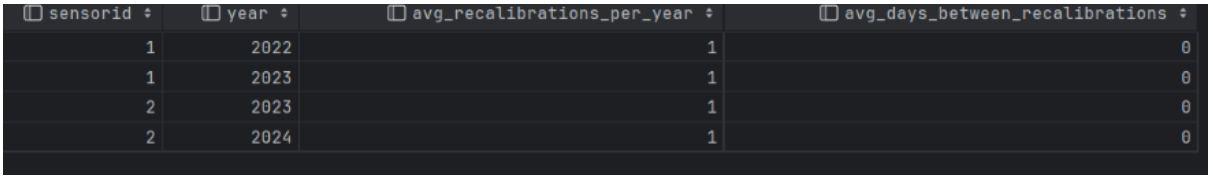
1,Requete 1 :

```
--regute 1 le nombre moyen de réétalonnages par an (déduisez-en le temps moyen entre deux réétalonages),
SELECT sensorid,
       EXTRACT(YEAR FROM timestamp) AS year,
       COUNT(timestamp) / COUNT(DISTINCT EXTRACT(YEAR FROM timestamp)) AS avg_recalibrations_per_year,
       EXTRACT(day FROM MAX(timestamp) - MIN(timestamp)) / count(timestamp) AS avg_days_between_recalibrations
FROM sensorhistory
WHERE status = 'recalibrated'
GROUP BY sensorid,year;
```

Explication de la requête 1 :

1. Extraction de l'année :
    - EXTRACT(YEAR FROM timestamp) : Cette fonction extrait l'année de chaque entrée de timestamp.
2. Calcul du nombre moyen de réétalonnages par an :
    - COUNT(timestamp) / COUNT(DISTINCT EXTRACT(YEAR FROM timestamp)) : On compte le nombre de timestamps où le statut est 'recalibrated' et on divise par le nombre d'années distinctes pour obtenir le nombre moyen de réétalonnages par an.
3. Calcul de la durée moyenne entre deux réétalonnages :
    - EXTRACT(day FROM MAX(timestamp) - MIN(timestamp)) / count(timestamp) : On calcule la différence entre les dates de réétalonnages maximales et minimales pour obtenir la durée totale, puis on divise par le nombre total de réétalonnages pour obtenir la durée moyenne entre deux réétalonnages.
4. Groupement par capteur et par année :
    - GROUP BY sensorid, year : On regroupe les résultats par ID de capteur et par année pour obtenir des résultats spécifiques à chaque capteur pour chaque année.

Resultat 1:



sensorid	year	avg_recalibrations_per_year	avg_days_between_recalibrations
1	2022	1	0
1	2023	1	0
2	2023	1	0
2	2024	1	0

Jeux de Test requête 1:

```sql
--Jeux de test
INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES

    (1, 'recalibrated', '2022-06-01 10:00:00'),
    (1, 'recalibrated', '2023-05-15 10:00:00')
;

INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES

    (2, 'recalibrated', '2023-06-01 10:00:00'),
    (2, 'recalibrated', '2024-05-15 10:00:00')
;
```

Voici mon nouveau résultat. Ma requête fonctionne bien, elle permet de montrer le nombre moyen de réétalonnages par an (ce qui permet également de déduire le temps moyen entre deux réétalonnages).

	sensorid	year	avg_recalibrations_per_year	avg_days_between_recalibrations
1	1	2022	2	0
2	1	2023	2	67
3	2	2023	2	15
4	2	2024	2	67.5

.

2,Requête 2 : la proportion de ceux qui n'ont pas été mis au rebut un an après leur mise en service (taux de survie à un an),

```
leur mise en service (taux de survie à un an),
WITH ServiceStart AS (
    SELECT
        sensorid,
        MIN(timestamp) AS service_start
    FROM
        SensorHistory
    WHERE
        status = 'in service'
    GROUP BY
        sensorid
),
     SurvivalRates AS (
         SELECT
             s.modelid,
             ss.sensorid,
             COUNT(*) FILTER (WHERE sh.timestamp <= ss.service_start +
INTERVAL '1 year' AND sh.status = 'decommissioned') = 0 AS
survived_one_year
         FROM
             ServiceStart ss
                 JOIN
             SensorHistory sh ON ss.sensorid = sh.sensorid
                 JOIN
             Sensor s ON ss.sensorid = s.sensorid
         GROUP BY
             s.modelid, ss.sensorid, ss.service_start
     )
SELECT
    s.sensorid,
    s.modelid,
    AVG(survived_one_year::int) * 100 AS one_year_survival_rate
FROM
    SurvivalRates sr
        JOIN
    Sensor s ON sr.sensorid = s.sensorid
GROUP BY
    s.sensorid, s.modelid;
```

Explication du requete 2:

1. Détermination de la date de mise en service :
    - WITH ServiceStart AS (SELECT sensorid, MIN(timestamp) AS service_start FROM SensorHistory WHERE status = 'in service' GROUP BY sensorid) : On crée une sous-requête pour obtenir la date de mise en service (la plus ancienne) pour chaque capteur.
2. Calcul du taux de survie à un an :

- COUNT(*) FILTER (WHERE sh.timestamp <= ss.service_start + INTERVAL '1 year' AND sh.status = 'decommissioned') = 0 AS survived_one_year : On compte les enregistrements où le capteur a été mis hors service dans l'année suivant sa mise en service. Si ce nombre est 0, cela signifie que le capteur a survécu un an.

3. Groupement par modèle et capteur :
    - GROUP BY s.modelid, ss.sensorid, ss.service_start : On regroupe les résultats par ID de modèle et par ID de capteur pour chaque date de mise en service.
4. Calcul du taux de survie :
    - AVG(survived_one_year::int) * 100 AS one_year_survival_rate : On calcule la moyenne des capteurs ayant survécu un an et on la multiplie par 100 pour obtenir un pourcentage.

Resultat:

	sensorid	modelid	one_year_survival_rate
1	1	5	100
2	2	6	100
3	4	12	100
4	5	14	100
5	6	15	100

Jeux de Test requête 2::

```sql
--Jeu de test requete 2
--cas 1 avant 1 ans
INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES

    (1, 'decommissioned', '2022-06-01 10:00:00');

--cas 2 apres plus 1 an

INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES

    (2, 'decommissioned', '2026-06-01 10:00:00');
```

résultat attendu : cas 1 changer mais cas 2 non

sensorid	modelid	one_year_survival_rate
1	5	0
2	6	100
4	12	100
5	14	100
6	15	100

Donc ca marche comme résultat attendu

3,Requête 3 :Proportion de capteurs non réétalonnés et non mis au rebut un mois après mise en service

```sql
WITH ServiceStart AS (
    SELECT
        sensorid,
        MIN(timestamp) AS service_start
    FROM
        SensorHistory
    WHERE
        status = 'in service'
    GROUP BY
        sensorid
),
    OneMonthCheck AS (
        SELECT
            s.modelid,
            ss.sensorid,
            COUNT(*) FILTER (WHERE sh.timestamp <= ss.service_start +
INTERVAL '1 month' AND sh.status IN ('recalibrated', 'decommissioned')) = 0
AS intact_after_one_month
        FROM
            ServiceStart ss
                JOIN
            SensorHistory sh ON ss.sensorid = sh.sensorid
                JOIN
            Sensor s ON ss.sensorid = s.sensorid
        GROUP BY
            s.modelid, ss.sensorid
    )
SELECT
    modelid,
    sensorid,
    AVG(intact_after_one_month::int) * 100 AS one_month_intact_rate
FROM
    OneMonthCheck
GROUP BY
    modelid, sensorid;
```

Explication de requête 3 :

:Détermination de la date de mise en service :

- WITH ServiceStart AS (SELECT sensorid, MIN(timestamp) AS service_start FROM SensorHistory WHERE status = 'in service' GROUP BY sensorid) : On crée une sous-requête pour obtenir la date de mise en service (la plus ancienne) pour chaque capteur.

2. Calcul du taux de survie à un an :
    - COUNT(*) FILTER (WHERE sh.timestamp <= ss.service_start + INTERVAL '1 year' AND sh.status = 'decommissioned') = 0 AS survived_one_year : On compte les enregistrements où le capteur a été mis hors service dans l'année suivant sa mise en service. Si ce nombre est 0, cela signifie que le capteur a survécu un an.

3. Groupement par modèle et capteur :
    - GROUP BY s.modelid, ss.sensorid, ss.service_start : On regroupe les résultats par ID de modèle et par ID de capteur pour chaque date de mise en service.
4. Calcul du taux de survie :
    - AVG(survived_one_year::int) * 100 AS one_year_survival_rate : On calcule la moyenne des capteurs ayant survécu un an et on la multiplie par 100 pour obtenir un pourcentage.

Jeux De Test Requête 3:

```sql
--Jeux de Test
-- cas tous les capteur cest bon apres un mois
INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES
    (4, 'in service', '2022-01-01 10:00:00'),
    (4, 'in service', '2024-01-01 10:00:00'),
    (5, 'in service', '2022-01-01 10:00:00'),
    (5, 'recalibrated', '2024-01-01 10:00:00');

--cas en revanche
INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES
    (6, 'in service', '2023-01-01 10:00:00'),
    (6, 'recalibrated', '2023-01-02 10:00:00');
```

Resultat:

modelid	sensorid	one_month_intact_rate
6	2	100
5	1	100
15	6	0
14	5	100
12	4	100

VOICI C'EST MON BDD: et Les requêtes(en text)

```
DROP TABLE IF EXISTS ControlMeasurement, SensorMeasurement, Controller,
Sensor, Model CASCADE;

CREATE TABLE Model(
   modelid int NOT NULL PRIMARY KEY,
   brand text,
   model text,
   name text,
   unit text
);

CREATE TABLE Sensor(
   sensorid int NOT NULL PRIMARY KEY UNIQUE ,
   position text,
   validity text,
   modelid int REFERENCES Model(modelid) UNIQUE ,
   serialnumber text UNIQUE
);

CREATE TABLE Controller(
   controllerid int NOT NULL PRIMARY KEY,
   modelid int NOT NULL REFERENCES Model(modelid),
   serialnumber text,
   UNIQUE (serialnumber)
);

CREATE TABLE SensorMeasurement(
   sensorid int NOT NULL REFERENCES Sensor(sensorid),
   timestamp TIMESTAMP UNIQUE ,
   sensorvalue double precision,
   PRIMARY KEY (sensorid, timestamp)
);

CREATE TABLE ControlMeasurement(
   controllerid int NOT NULL references Controller(controllerid),
   sensorid int NOT NULL REFERENCES Sensor(sensorid),
   sensortimestamp TIMESTAMP references SensorMeasurement(timestamp),
   controltimestamp TIMESTAMP UNIQUE ,
   controlvalue double precision,
   PRIMARY KEY (controllerid, controltimestamp)

);

CREATE TABLE SensorHistory (
   sensorHistoryId SERIAL PRIMARY KEY,
   sensorid INT REFERENCES Sensor(sensorid),
   status TEXT CHECK (status IN ('in service', 'recalibrated',
'decommissioned')),
   timestamp TIMESTAMP
```

```
);

insert into public.model (modelid, brand, model, name, unit)
values
   (1, 'TWK', 'IW 150', 'Inductive linear transducer', 'mm'),
   (2, 'PCB', 'M352C68', 'ICP miniature vibration sensor', 'Hz'),
   (3, 'TWK', 'PW6C', 'Single point load cell', 'kg'),
   (4, 'TWK', 'PW2C', 'Single point load cell', 'kg'),
   (5, 'PCB', 'M353B18', 'Accelerometer', 'm·s⁻²'),
   (6, 'PCB', '40PC150G2A', 'Pressure sensor', 'psi'),
   (7, 'PCB', '208A03', 'Force sensor', 'kg'),
   (8, 'Foo', 'Bar', 'Portable pressure sensor', 'psi'),
   (9, 'Foo', 'Bar', 'Portable acceleration sensor', 'm·s⁻²'),
   (10, 'NIN', 'PHC', 'pH sensor', ''),
   (11, 'NIN', 'PBD', 'Portable pH sensor', ''),
   (12, 'NIN', 'Heat42', 'Heat sensor', '°C'),
   (13, 'NIN', 'M4R10', 'Thermometer', '°C'),
   (14, 'PCE', 'WNM-100', 'Whiteness sensor', ''),
   (15, 'NIN', 'LU161', 'Moisture sensor', ''),
   (16, 'NIN', 'T04D', 'Speed sensor', 'm·s⁻¹');


insert into public.controller (controllerid, modelid, serialnumber)
values  (1, 9, '9873'),
       (2, 8, '379134'),
       (3, 11, '99732'),
       (4, 13, '13001'),
       (5, 14, '77126'),
       (6, 15, '4242'),
       (7, 16, '23234');

insert into public.sensor (sensorid, position, validity, modelid,
serialnumber)
values  (1, 'Breast roller acceleration', '0 years 0 mons 0 days 0 hours 0
mins 5.0 secs', 5, '136031'),
       (2, 'Headbox pressure', '0 years 0 mons 0 days 0 hours 1 mins 0.0
secs', 6, '194432'),
       (3, 'Headbox pH', '0 years 0 mons 0 days 0 hours 30 mins 0.0 secs',
10, '523133'),
       (4, 'Heated dryer temperature', '0 years 0 mons 0 days 0 hours 10 mins
0.0 secs', 12, '451874'),
       (5, 'Wet press whiteness', '0 years 0 mons 0 days 0 hours 10 mins 0.0
secs', 14, '98234'),
       (6, 'Wet press moisture', '0 years 0 mons 0 days 1 hours 0 mins 0.0
secs', 15, '424242'),
       (7, 'Wire mesh speed', '0 years 0 mons 0 days 0 hours 0 mins 5.0
secs', 16, '98734');
-- Example data for SensorHistory
INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES
   (1, 'in service', '2022-01-01 10:00:00'),
   (1, 'recalibrated', '2022-06-01 10:00:00'),
   (1, 'recalibrated', '2023-01-01 10:00:00'),
   (1, 'decommissioned', '2024-01-01 10:00:00'),
```

```
   (2, 'in service', '2023-01-01 10:00:00'),
   (2, 'recalibrated', '2023-07-01 10:00:00'),
   (2, 'recalibrated', '2024-01-01 10:00:00');



--Jeux de Test
INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES
   (4, 'in service', '2022-01-01 10:00:00'),
   (4, 'in service', '2024-01-01 10:00:00');



--Jeux de Test de requete 2 il faut attention 3 cas  posibilite arriver
-- cas1 cest la duration dure < un an apres en service " donc ca va pas
affihcher
-- cas2 cest la duration dure plus que un a apres en service
-- cas3  la duration dure equal un an apres en service (probalite :rare)
--cas4 la status cest "en panne" ou "fixer"



-- 3 Proportion de capteurs non réétalonnés et non mis au rebut un mois après
mise en service


--Jeux de Test pour dernier requete
--cas 1  status "fixer" ou "en panne"
--cas 2 <1 month et status "in service"
--cas 3 = moth status "in service" (probilite = rare)
--cas 4 > 1 motnh et satus "in service"









--test


--requte 1 le nombre moyen de réétalonnages par an (déduisez-en le temps
moyen entre deux réétalonages),
SELECT sensorid,
      EXTRACT(YEAR FROM timestamp) AS year,
      COUNT(timestamp) / COUNT(DISTINCT EXTRACT(YEAR FROM timestamp)) AS
avg_recalibrations_per_year,
      EXTRACT(day FROM MAX(timestamp) - MIN(timestamp)) / count(timestamp) AS
avg_days_between_recalibrations
FROM sensorhistory
WHERE status = 'recalibrated'
GROUP BY sensorid,year;
```

```sql
--Jeux de test
INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES

   (1, 'recalibrated', '2022-06-01 10:00:00'),
   (1, 'recalibrated', '2023-05-15 10:00:00')
;

INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES

   (2, 'recalibrated', '2023-06-01 10:00:00'),
   (2, 'recalibrated', '2024-05-15 10:00:00')
;

INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES

   (1, 'recalibrated', '2022-08-01 10:00:00')

;




-- requete 2
WITH ServiceStart AS (
   SELECT
       sensorid,
       MIN(timestamp) AS service_start
   FROM
       SensorHistory
   WHERE
       status = 'in service'
   GROUP BY
       sensorid
),
    SurvivalRates AS (
        SELECT
            s.modelid,
            ss.sensorid,
            COUNT(*) FILTER (WHERE sh.timestamp <= ss.service_start +
INTERVAL '1 year' AND sh.status = 'decommissioned') = 0 AS survived_one_year
        FROM
            ServiceStart ss
                JOIN
            SensorHistory sh ON ss.sensorid = sh.sensorid
                JOIN
            Sensor s ON ss.sensorid = s.sensorid
        GROUP BY
            s.modelid, ss.sensorid, ss.service_start
    )
SELECT
   s.sensorid,
```

```
    s.modelid,
    AVG(survived_one_year::int) * 100 AS one_year_survival_rate
FROM
    SurvivalRates sr
        JOIN
    Sensor s ON sr.sensorid = s.sensorid
GROUP BY
    s.sensorid, s.modelid;
--Jeu de test requete 2
--cas 1 avant 1 ans
INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES

    (1, 'decommissioned', '2022-06-01 10:00:00');

--cas 2 apres plus 1 an

INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES

    (2, 'decommissioned', '2026-06-01 10:00:00');


--requte 3
WITH ServiceStart AS (
    SELECT
        sensorid,
        MIN(timestamp) AS service_start
    FROM
        SensorHistory
    WHERE
        status = 'in service'
    GROUP BY
        sensorid
),
     OneMonthCheck AS (
         SELECT
             s.modelid,
             ss.sensorid,
             COUNT(*) FILTER (WHERE sh.timestamp <= ss.service_start +
INTERVAL '1 month' AND sh.status IN ('recalibrated', 'decommissioned')) = 0
AS intact_after_one_month
         FROM
             ServiceStart ss
                 JOIN
             SensorHistory sh ON ss.sensorid = sh.sensorid
                 JOIN
             Sensor s ON ss.sensorid = s.sensorid
         GROUP BY
             s.modelid, ss.sensorid
     )
SELECT
    modelid,
    sensorid,
    AVG(intact_after_one_month::int) * 100 AS one_month_intact_rate
```

```
FROM
   OneMonthCheck
GROUP BY
   modelid, sensorid;

--Jeux de Test
-- cas tous les capteur cest bon apres un mois
INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES
   (4, 'in service', '2022-01-01 10:00:00'),
   (4, 'in service', '2024-01-01 10:00:00');

INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES
   (4, 'recalibrated', '2022-01-01 10:00:00'),
   (4, 'in service', '2024-01-03 10:00:00');



--cas en revanche
INSERT INTO SensorHistory (sensorid, status, timestamp) VALUES
   (6, 'in service', '2023-01-01 10:00:00'),
   (6, 'recalibrated', '2023-01-02 10:00:00');
```