

# Projet Base de Données

QUIVRON Loïc, ALONSO ALONSO  
David, LENG Mathieu



# Hypothèses de la DB

L'épidémiologiste peut  
tout modifier

L'utilisateur lambda peut  
juste voir toutes les  
données

Un seul épidémiologiste  
encode une statistique  
journalière (mais  
adaptation pour  
l'exercice)

Pourraient encoder les  
stats à différents  
moments

- plusieurs tables
- pas forcément NOT NULL

Hosp\_patients >  
icu\_patients

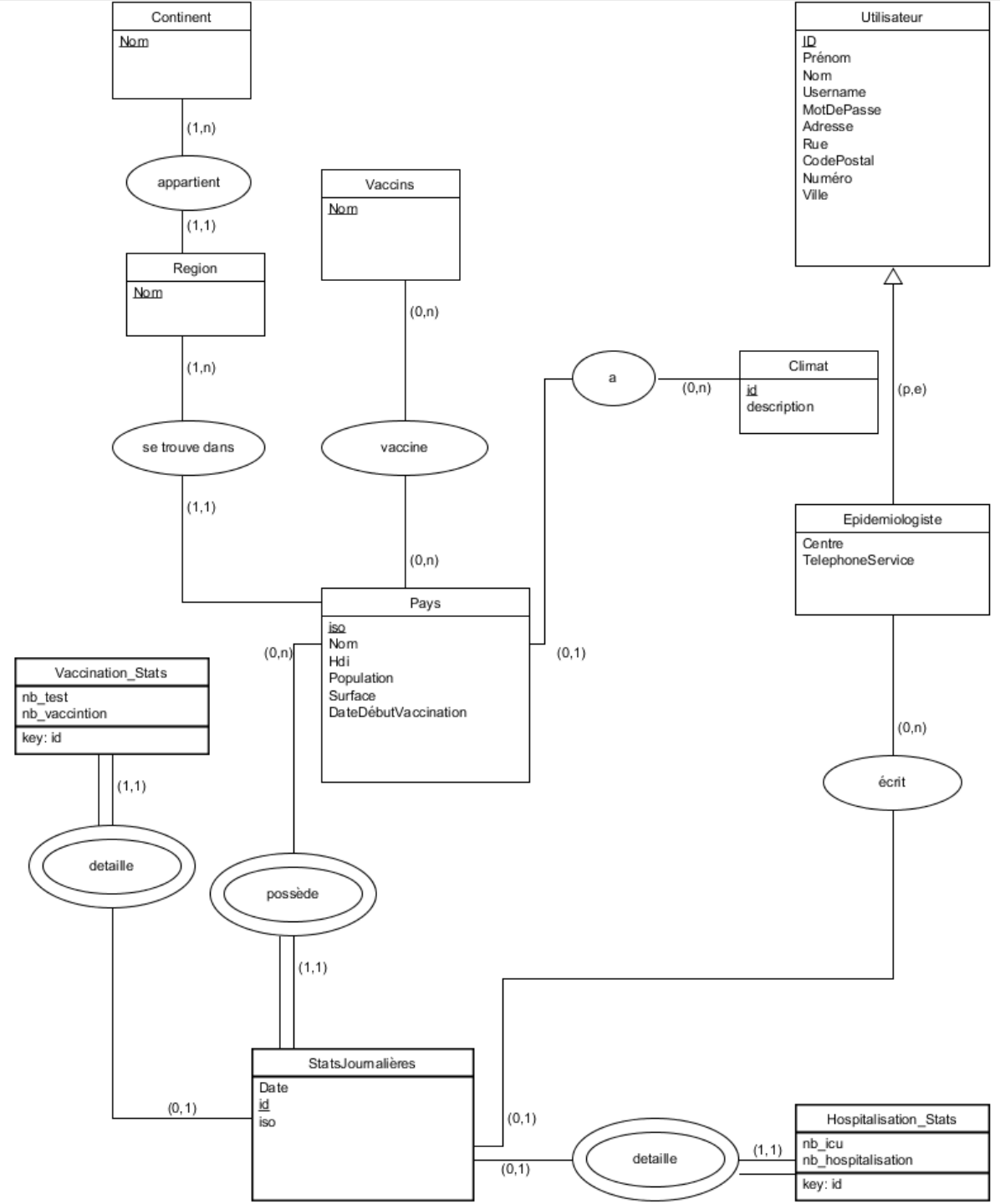
Pseudo unique

Adresse physique

NB\_vaccins, tests, hosp,  
peuvent être >  
population

# Modèle entité-association

- Les hypothèses
- Les contraintes d'intégrité
  - Adresse entière
  - $Nb\_icu \leq nb\_hospitalisation$
  - $DateDébutVaccination \leq Date$
  - ...



# Modèle relationnel

## Les contraintes d'intégrité

- Region.nom\_continent est unique
- Pays.nom est unique

- Continent(nom)
- Region(nom, nom\_continent)  
Region.nom\_continent référence Continent.nom
- Pays(iso, nom, hdi, population, surface, region, climat, debut\_vaccination)  
Pays.region référence Region.nom  
Pays.climat référence Climat.id
- Climat(id, description)

- Vaccins(nom)
- Campagne\_Vaccin(nom\_vaccin, iso\_pays)  
Campagne\_Vaccin.nom\_vaccin référence Vaccin.nom  
Campagne\_Vaccin.iso\_pays référence Pays.iso

# Modèle relationnel

## Les contraintes d'intégrité

- Hospitalisations\_Stats.icu\_patients <= Hospitalisations\_Stats.hosp\_patients
- Utilisateur.pseudo est unique
- Utilisateur.uuid est unique

- Stats\_Journalieres(id, iso\_pays, date, id\_epi)  
Stats\_Journalieres.iso\_pays référence Pays.iso  
Stats\_Journalieres.id\_epi référence Epidemiologiste.uuid
- Vaccinations\_Stats(id, test, vaccination)  
Vaccinations\_Stats.id référence Stats\_Journalieres.id
- Hospitalisations\_Stats(id, icu\_patients, hosp\_patients)  
hospitalisations.id référence Stats\_Journalieres.id

- Utilisateur(id, uuid, prenom, nom, pseudo, mot\_de\_passe, rue\_adresse, code\_postal\_adresse, numero\_adresse, ville\_adresse)
- Epidemiologiste(uuid, centre, telephone\_service)  
Epidemiologiste.uuid référence utilisateur.uuid



# Nettoyage des données

- CSV\_to\_Table.py
  - Script python
  - Plus lent que SQL
  - Mais fonctionne pour tous SGBD
- Convert ugly date to pretty date
- Récupérer la date de début vacc dans producers et l'ajouter dans pays
- Générer des id pour les stats journalières
- Séparer les régions des continents,
- Etc...

```
def convert_ugly_date_to_pretty_date(ugly_date_string):  
    dto = parser.parse(ugly_date_string)  
    return dto.strftime("%d/%m/%Y")
```

```
CREATE TABLE Utilisateur (  
    id SERIAL NOT NULL PRIMARY KEY,  
    uuid uuid NOT NULL UNIQUE,  
    Nom varchar(40),  
    Prenom varchar(40),  
    pseudo varchar(40) UNIQUE , --should be not null  
    mot_de_passe varchar(100) , --should be not null  
    rue_Adresse varchar(100),  
    code_postal_adresse INT,  
    numero_adresse INT,  
    ville_adresse varchar(40)  
    CONSTRAINT adresse_integrity_constraint CHECK (((rue_adresse is NULL)::int + (rue_adr  
    |+(code_postal_adresse is NULL)::int + (numero_adresse is NULL)::int )in (0,4))
```

Implémentation des contraintes  
d'intégrité |

# Implémentation des contraintes d'intégrité

```
CREATE OR REPLACE FUNCTION auto_insert_pays() RETURNS TRIGGER  
LANGUAGE PLPGSQL AS $$  
BEGIN
```

```
    IF NOT EXISTS (SELECT iso FROM pays WHERE iso=NEW.iso_pays) THEN  
        INSERT INTO pays(iso) VALUES (NEW.iso_pays);  
    END IF;  
    RETURN NEW;  
END; $$;
```

```
CREATE TRIGGER auto_insert_pays BEFORE INSERT ON campagne_vaccin FOR EACH ROW
```

```
CREATE TRIGGER stats_journalieres_date_after_start_campaign  
BEFORE INSERT OR UPDATE OF vaccinations ON vaccinnations_stats FOR EACH ROW  
EXECUTE PROCEDURE stats_journalieres_date_after_start_campaign();
```



# Implémentation des contraintes d'intégrité

```
CREATE OR REPLACE FUNCTION stats_journalieres_date_after_start_campaign() RETURNS TRIGGER
LANGUAGE PLPGSQL AS $$
DECLARE
    date_vs stats_journalieres.date%type;
    date_start_campaign pays.debut_vaccination%type;
BEGIN
    SELECT date
    FROM stats_journalieres s
    INTO date_vs
    WHERE s.id=NEW.id;
    raise notice 'Date VS : %s', date_vs;

    SELECT debut_vaccination
    FROM pays p
    JOIN stats_journalieres s ON p.iso=s.iso_pays
    INTO date_start_campaign
    WHERE s.id=NEW.id;
    raise notice 'Date Start Campaign : %s', date_vs;

    IF(NEW.vaccinations is NOT NULL AND date_vs < date_start_campaign) THEN
        RAISE EXCEPTION 'La date de la stat journalière doit être postérieure à la date de début de vaccination du pays concerné';
    END IF;
    RETURN NEW;
END; $$;
```

# Requêtes

- 6 Requêtes
  1. Hospitalisations  $\geq 5000$
  2. Plus gros vaccinateur
  3. Qui utilise quel vaccin
  4. % hospitalisé, le 01/01/2021
  5. Evolution hospitalisations
  6. Vaccin commun France-Belgique



# Requête 1

Sélectionnez les pays qui, au même moment, ont eu au moins 5000 personnes hospitalisées(**hosp\_patients**).

SQL

```
SELECT distinct p.iso, p.nom
FROM pays p JOIN stats_journalieres sj ON p.iso=sj.iso_pays JOIN hospitalisations_stats hs ON hs.id=sj.id
WHERE hs.hosp_patients >= 5000
```

Algèbre  
relationnel

$$\begin{aligned} \text{HOSP\_STATS\_JOURN} &\leftarrow \text{hospitalisations\_stats} * \text{stats\_journalieres} \\ \text{BIG\_HOSP} &\leftarrow \sigma_{\text{hosp\_patients} \geq 5000}(\text{HOSP\_STATS\_JOURN}) \\ \text{BIG\_HOSP\_PAYS} &\leftarrow \text{Pays} *_{I \text{ so} = \text{Iso\_Pays}} \text{stats\_journalieres} \\ &\pi_{iso, nom}(\text{BIG\_HOSP\_PAYS}) \end{aligned}$$

Calcul  
Tuple

$$\{\text{pays.iso}, \text{pays.nom} \mid \text{Pays}(\text{pays}) \cap \exists sj (\text{stats\_journalieres}(sj) \cap \text{hospitalisations\_stats}(hs) \cap sj.id = hs.id \cap sj.iso\_pays = \text{pays.iso} \cap hs.hosp\_patients \geq 5000)\}$$

# Requête 2

Sélectionnez le pays qui a administré le plus grand nombre total de vaccins (toutes les dates cumulées)

```
WITH sum_vacc_for_each_pays(iso,nom, somme ) AS (  
    SELECT distinct p.iso, p.nom ,SUM(vs.vaccinations)  
    FROM pays p  
    JOIN stats_journalieres sj ON p.iso = sj.iso_pays  
    JOIN vaccinations_stats vs ON vs.id=sj.id  
    GROUP BY p.nom, p.iso  
)  
select iso,nom, somme  
from sum_vacc_for_each_pays maxi  
WHERE somme is not null  
ORDER BY somme DESC limit 1
```

# Requête 3

Pour chaque vaccin, sélectionnez le nom des pays qui l'utilisent

```
SELECT distinct cv.nom_vaccin, p.iso, p.nom  
FROM pays p JOIN campagne_vaccin cv ON p.iso=cv.iso_pays  
GROUP BY cv.nom_vaccin, p.iso, p.nom  
order by cv.nom_vaccin
```

# Requête 4

La proportion de la population hospitalisée pour chaque pays, le 1er janvier 2021

```
SELECT p.iso, p.nom, (( hs.hosp_patiens::float/ p.population )*100)::numeric(36,2) as pourcentage
FROM hospitalisations_stats hs JOIN stats_journalieres sj ON sj.id = hs.id JOIN pays p ON p.iso= sj.iso_pays
WHERE sj.date = '01/01/2021'
Group by p.iso, p.population, hs.hosp_patiens, p.nom
```



# Requête 5

Calculez l'évolution, pour chaque jour et chaque pays, du nombre de patients hospitalisés (**hosp\_patients**)

```
SELECT p.iso, p.nom, sj.date, hs.hosp_patients - LAG(hs.hosp_patients)OVER (PARTITION BY p.iso ORDER BY sj.date) evolution
FROM hospitalisations_stats hs JOIN stats_journalieres sj on sj.id = hs.id join pays p on p.iso = sj.iso_pays
Group by p.iso, p.nom, sj.date, hs.hosp_patients
order by p.iso, sj.date
```

# Requête 6

Sélectionnez le nom des vaccins disponibles à la fois en Belgique et en France

SQL

```
SELECT cv.nom_vaccin
FROM campagne_vaccin cv
WHERE iso_pays = 'BEL'
INTERSECT
SELECT cv.nom_vaccin
FROM campagne_vaccin cv
WHERE iso_pays = 'FRA'
```

Algèbre  
relationnel

$$\begin{aligned} BEL &\leftarrow \sigma_{iso\_pays='BEL'}(campagne\_vaccin) \\ FRA &\leftarrow \sigma_{iso\_pays='FRA'}(campagne\_vaccin) \\ \pi_{nom\_vaccin}(BEL \cap FRA) \end{aligned}$$

Calcul  
Tuple

$$\{cv.nom \mid campagne\_vaccin(cv) \cap \exists cv2(campagne\_vaccin(cv2) \cap cv2.nom\_vaccin = cv.nom\_vaccin \cap cv2.iso\_pays = 'BEL' \cap cv.iso\_pays = 'FRA')\}$$

# Adaptations

- Certaines données censées être non nulles
  - Ex nom des pays, pseudo mot de passe utilisateur
- OWID\_WRL représente le monde entier, donc il fausse le calcul
- Utilisation d'un id numérique pour l'utilisateur afin de gérer la connexion avec Django

# Plus ... toujours plus !

- Implémentation de trigger
- Une GUI très propre
- Formulaire utilisateur
- Des boutons... Et un formulaire de requêtes
  - Gestion des erreurs en 'AJAX'
  - Feedback précis et ergonomique
- Vue de profil
- Migrations et reverse migrations

# Faibles/A améliorer

- Vérification du type d'instruction
- Utilisation du DCL pour gérer les accès
  - 2 connexions
- Trigger modification de date
- 403 Forbidden Access
- Réponse requêtes en AJAX