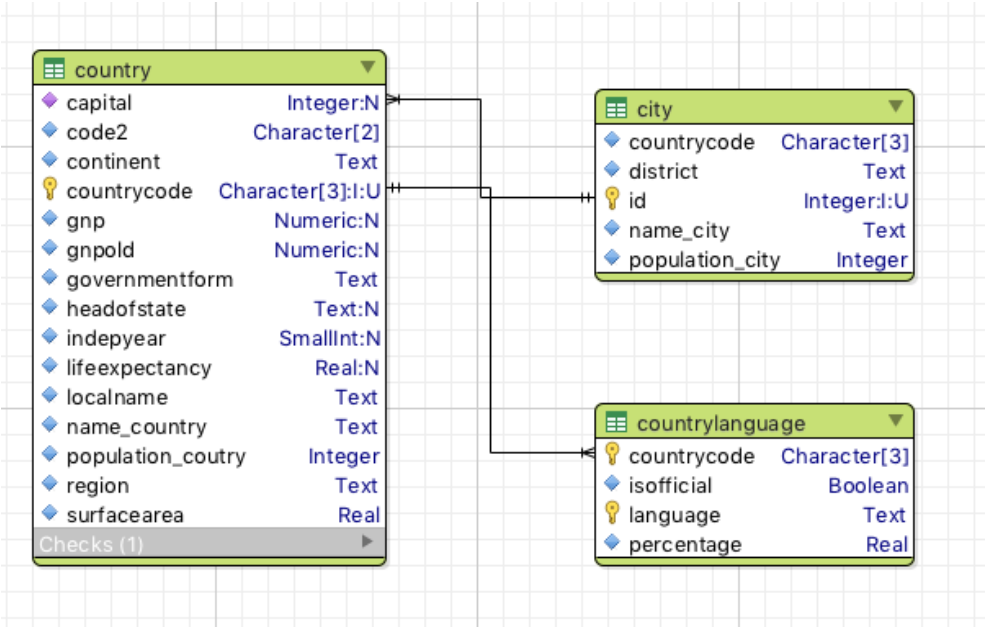
Le sujet comprend 4 exercices. Tous les exercices sont à faire sur le sujet.

On utilisera le schéma `world` ci-dessous dans les deux premiers exercices. L'attribut `countrycode` est la clef primaire de la table `country`. L'attribut `id` est la clef primaire de la table `city`. Les deux attributs `countrycode` et `language` forment la clef primaire de la table `countrylanguage`. Les clefs étrangères sont de la forme :

— `Countrylanguage(countrycode)` ⊆ `Country(countrycode)`
— `Country(capital)` ⊆ `City(id)`



1 Exercice : écriture de requêtes

1) Écrire une requête en algèbre relationnelle retournant les pays (`name_country`) dont l'une des langues officielles est le Français (`french`).

2) Écrire une requête en algèbre relationnelle retournant les pays (`countrycode`) qui n'ont pas de langue officielle.

3) Écrire une requête en SQL retournant les pays (`countrycode`) dont toutes les langues sont officielles.

4) Écrire une requête en SQL donnant pour chaque continent la langue la plus parlée.

2 QCM

Pour toutes les questions, cochez **toutes les requêtes** dont le résultat contient **exactement** la réponse à la question. Il y a toujours au moins une solution. Le QCM (c'est à dire, la question 2 de l'examen) ne pourra pas donner lieu à des points négatifs (la note minimum pour l'exercice sera de 0). En revanche, les réponses fausses à une question seront pénalisées.

1) Parmi les requêtes suivantes, laquelle ou lesquelles calculent :
les noms des pays où plus de 2 villes ont plus de 5 millions d'habitants

☐

```
WITH CC as (SELECT countrycode FROM city NATURAL JOIN country GROUP BY
    countrycode HAVING COUNT(name_city)>=2 and SUM(population_city) >=
    5000000 )
SELECT name_country FROM country NATURAL JOIN CC;
```

☒

```
WITH CC as (SELECT DISTINCT c1.countrycode FROM city as c1, city as c2
    WHERE c1.countrycode=c2.countrycode AND c1.id!=c2.id AND c1.
    population_city>=5000000 AND c2.population_city>=5000000) SELECT
    name_country FROM country NATURAL JOIN CC;
```

☒

```
WITH CC as (SELECT countrycode FROM city NATURAL JOIN country WHERE
    population_city >= 5000000 GROUP BY countrycode HAVING COUNT(
    name_city)>=2 )
SELECT name_country FROM country NATURAL JOIN CC;
```

2) Parmi les requêtes suivantes, laquelle ou lesquelles calculent :
le continent où l'espérance de vie moyenne est la plus grande.

☒

```
with R as(select continent, sum(population_country*lifeexpectancy)/sum(
  population_country) as espv from country group by continent),
  S as (select max(espv) from R)
select continent from R where espv=all(select * from S);
```

☒

```
with R as(select continent, sum(population_country*lifeexpectancy)/sum(
  population_country) as espv from country group by continent),
  S as (select max(espv) from R)
select continent from R where espv=any(select * from S);
```

☒

```
with R as(select continent, sum(population_country*lifeexpectancy)/sum(
  population_country) as espv from country group by continent),
  S as (select max(espv) from R)
select continent from R where espv in (select * from S);
```

☐

```
with R as(select continent, sum(population_country*lifeexpectancy)/sum(
  population_country) as espv from country group by continent order by
  espv),
S as (select continent, max(espv) from R group by continent)
select continent from S;
```

3) Parmi les requêtes suivantes, laquelle ou lesquelles calculent :
Le noms des villes qui apparaissent dans au moins deux pays.

☒

```
select name_city from city
    group by name_city having count(countrycode)>=2;
```

☐

```
select name_city , count(countrycode) from city
    group by name_city where count >=2;
```

☒

```
select a.name_city from city as a, city as b
    where a.name_city = b.name_city and a.countrycode != b.countrycode;
```

☐

```
select a.name_city from city as a, city as b
    where a.name_city = b.name_city;
```

3 Exercice : requêtes récursives

Calculez le résultat des deux requêtes suivantes :

```
WITH RECURSIVE t(n) AS
( VALUES (1)
UNION
SELECT n+1 FROM t WHERE n < 5 )
SELECT sum(n) FROM t;
```

```
WITH RECURSIVE t(n) AS
( VALUES (1)
UNION
SELECT mod((n+1)::int,2) FROM t WHERE n < 5 )
SELECT sum(n) FROM t;
```

Extrait du manuel de Postgres : `VALUES` calcule une valeur de ligne ou un ensemble de valeurs de lignes spécifiées par des expressions. C'est généralement utilisé pour générer une « table statique » à l'intérieur d'une commande plus large mais elle peut aussi être utilisée séparément.

Quand plus d'une ligne est indiquée, toutes les lignes doivent avoir le même nombre d'éléments. Les types de données des colonnes de la table résultante sont déterminés en combinant les types explicites et les types inférés des expressions apparaissant dans cette colonne, en utilisant les mêmes règles que pour l'`UNION`.

À l'intérieur de grosses commandes, `VALUES` est autorisé au niveau de la syntaxe partout où la commande `SELECT` l'est. Comme la grammaire traite cette commande comme un `SELECT`, il est possible d'utiliser les clauses `ORDER BY`, `LIMIT` (ou de façon équivalente `FETCH FIRST`) et `OFFSET` avec une commande `VALUES`.

Remarque : `mod((n+1)::int,2)` retourne le reste de la division entière de `n+1` par `2` (`::` est un opérateur de conversion de type, si vous testez les deux dernières requêtes avec postgres, vous constaterez que la conversion en entier est nécessaire pour pouvoir utiliser ici la fonction `mod`).

4 Exercice : modélisation

On cherche à informatiser l'activité d'un opérateur téléphonique. L'opérateur possède des clients identifiés par un numéro de client. Pour chaque client, on connaît son nom, son prénom, son adresse de facturation, et sa date d'anniversaire. Chaque client possède un ou plusieurs abonnements téléphoniques (un par numéro de téléphone). Dans le cadre de chaque abonnement, le client a choisi des prestations (avec ou sans appels illimités, avec ou sans données internet,...), à une date donnée pour une durée fixée à l'avance. Chaque opérateur se doit de garder la trace de certaines informations liées aux appels téléphoniques dans le cadre de chaque abonnement. Chaque appel téléphonique est donc enregistré (numéro appelant, numéro appelé, date et heure de l'appel, durée).

1) Proposer une modélisation E/R pour décrire l'activité de cet opérateur. On précisera les entités, les clefs primaires, les associations, on indiquera pour chaque association les cardinalités. On spécifiera également les contraintes externes.

2) Proposer une traduction dans le modèle relationnel. On précisera les clefs primaires et les contraintes dont la syntaxe a été vue en cours (clefs étrangères, contraintes de non nullité, d'unicité, de vérification). Vous pouvez spécifier la structure des tables et les contraintes en français si vous ne vous souvenez plus de la syntaxe SQL (par exemple en spécifiant les clefs étrangères avec des inclusions, c.f. description du schéma `world` page 1 du sujet).