Penser les données par un ORM: implémentations et interactions

Maxime Challon -- EnC -- Atos

Lecture des données

query.all()

```python
# routes/generales.py /all

from ..models.factbook import Country

...
query =  Country.query
tous_resultats = query.all()
...
```

```sql
SELECT *
FROM country
```

query.first()

```
# routes/generales.py /first

from ..models.factbook import Country


...
query =  Country.query
resultat = query.first()
...
```

```
SELECT *
FROM country
LIMIT 1
```

query.limit()

```python
# routes/generales.py /limit

from ..models.factbook import Country

...
query =  Country.query
resultat = query.limit(10).all()
...
```

```sql
SELECT *
FROM country
LIMIT 10
```

query.get()

```python
# routes/generales.py /get

from ..models.factbook import Country

...
query =  Country.query
resultat = query.get("fr")
...
```

```sql
SELECT *
FROM country
WHERE id = 'fr'
```

Appliquer une condition avec `query.filter()`

```
query.filter(Classe_du_modele.champ
operateur_de_comparaison Valeur)
```

```
# routes/generales.py /filter

from ..models.factbook import Country

...
query =  Country.query
resultat = query.\
    filter(Country.type == "sovereign").\
    all()
...
```

```
SELECT *
FROM country
WHERE type = 'sovereign'
```

Appliquer des conditions : `AND` implicite

```python
# routes/generales.py /filter_and_implicite

from ..models.factbook import Country

...
query =  Country.query
resultat = query.\
    filter(Country.type == "sovereign",
         Country.id == 'fr').\
    all()
...
```

```sql
SELECT *
FROM country
WHERE type = 'sovereign' AND id = 'fr'
```

AND explicite

```
query.filter(and_(Classe_du_modele.champ operateur_de_comparaison Valeur, Classe_du_modele.champ operateur_de_comparaison Valeur))
```

```
# routes/generales.py /filter_and_explicite

from ..models.factbook import Country
from sqlalchemy import and_


...
query =  Country.query
resultat = query.\
    filter(and_(Country.type == "sovereign",
        Country.id == 'fr')).\
    all()
...
```

```
SELECT *
FROM country
WHERE type = 'sovereign' AND id = 'fr'
```

OR implicite

```
# routes/generales.py /filter_or_implicite

from ..models.factbook import Country

...
query =  Country.query
resultats = query.\
    filter((Country.type == 'dependency') |
        (Country.id == 'fr')).\
    all()
...
```

```
SELECT *
FROM country
WHERE type = 'dependency' OR id = 'fr'
```

OR explicite

```python
# routes/generales.py /filter_or_explicite

from ..models.factbook import Country
from sqlalchemy import or_


...
query =  Country.query
resultat = query.\
    filter(or_(Country.type == "dependency",
        Country.id == 'fr')).\
    all()
...
```

```sql
SELECT *
FROM country
WHERE type = 'dependency' OR id = 'fr'
```

Fonctions d'agrégation: `query.count()`

```
# routes/generales.py /count

from ..models.factbook import Country


...
query =  Country.query
resultat = query.\
    filter(Country.type == 'sovereign').\
    count()
...
```

```
SELECT COUNT(*)
FROM country
```

Tri ascendant : `query.order_by()`

```
# routes/generales.py /order_by

from ..models.factbook import Country


...
query =  Country.query
resultat = query.\
    order_by(Country.name).\
    all()
...
```

```
SELECT *
FROM country
ORDER BY name
```

Tri descendant : `query.order_by()`

```
# routes/generales.py /order_by_desc

from ..models.factbook import Country


...
query =  Country.query
resultat = query.\
    order_by(Country.name.desc()).\
    all()
...
```

```
SELECT *
FROM country
ORDER BY name DESC
```

query.group_by()

```
# routes/generales.py /group_by

from ..models.factbook import Country


...
query =  Country.query
resultat = query.\
    group_by(Country.type).\
    all()
...
```

```
SELECT type
FROM country
GROUP BY type
```

query.with_entities().distinct()

```
# routes/generales.py /distinct

from ..models.factbook import Country


...
query =  Country.query
resultat = query.\
    with_entities(Country.type).\
    distinct()
...
```

```
SELECT DISTINCT type
FROM country
```

query.having()

```
# routes/generales.py /having

from ..models.factbook import Country
from sqlalchemy import func


...
query =  Country.query
resultat = query.\
    group_by(Country.type).\
    having(
        func.count(Country.name) > 20
    ).\
    all()
...
```

```
SELECT type, COUNT(*)
FROM country
GROUP BY type
HAVING COUNT(*) > 20
```

Jointures: utilisation des relations entre classes de modèles

```
# models/factbook.py Country()
...
resources = db.relationship(
    'Resources',
    secondary=country_resources,
    backref="resources")
...
```

```
# models/factbook.py Country()
...
db.Column('resource', db.String(100), db.ForeignKey('resources.id'), primary_key=True)
...
```

Relations entre classes

```
from ..models.factbook import Country

...
query =  Country.query
resultat = query.all()
...
```

```
SELECT *
FROM country a
INNER JOIN country_resources b on b.id = a.id
INNER JOIN resources c on c.id = b.resource
```

TD: afficher les ressources d'un pays

Ecrire une route `/ressources/<string:nom_pays>` qui affcihe la liste des ressources possédées par le pays

- dans la route, filtrer les données sur le nom du pays
- retourner les données des ressources dans `pages/pays.html`
- effectuer une boucle Jinja2 sur les ressources

La route `/ressources/<string:nom_pays>`

```
# routes/generales.py /ressources

...
@app.route("/ressources/<string:nom>")
def ressources(nom):
    ressources = []

    query =  Country.query
    ressources = query.\
        filter(Country.name == nom).\
        first()

    return render_template("pages/pays.html",
        pays=nom, ressources=ressources, sous_titre=nom)
```

Le template `pages/pays.html`

```
<!-- templates/pages/pays.html -->

...
{%if ressources %}
    <ul>
        {%for resource in ressources.resources %}
            <li>{{resource.name}}</li>
        {%endfor%}
    </ul>
{%else%}
    <p>Il n'y a pas de ressources connues pour ce pays</p>
{%endif%}
...
```

Jointures: `query.select_from()` et `query.join()`

```
# routes/generales.py /altitude

...
donnees =  db.session.\
        query(Elevation).\
        select_from(Country).\
        join(Country.elevations).\
        filter(Country.name == nom).\
        all()
...
```

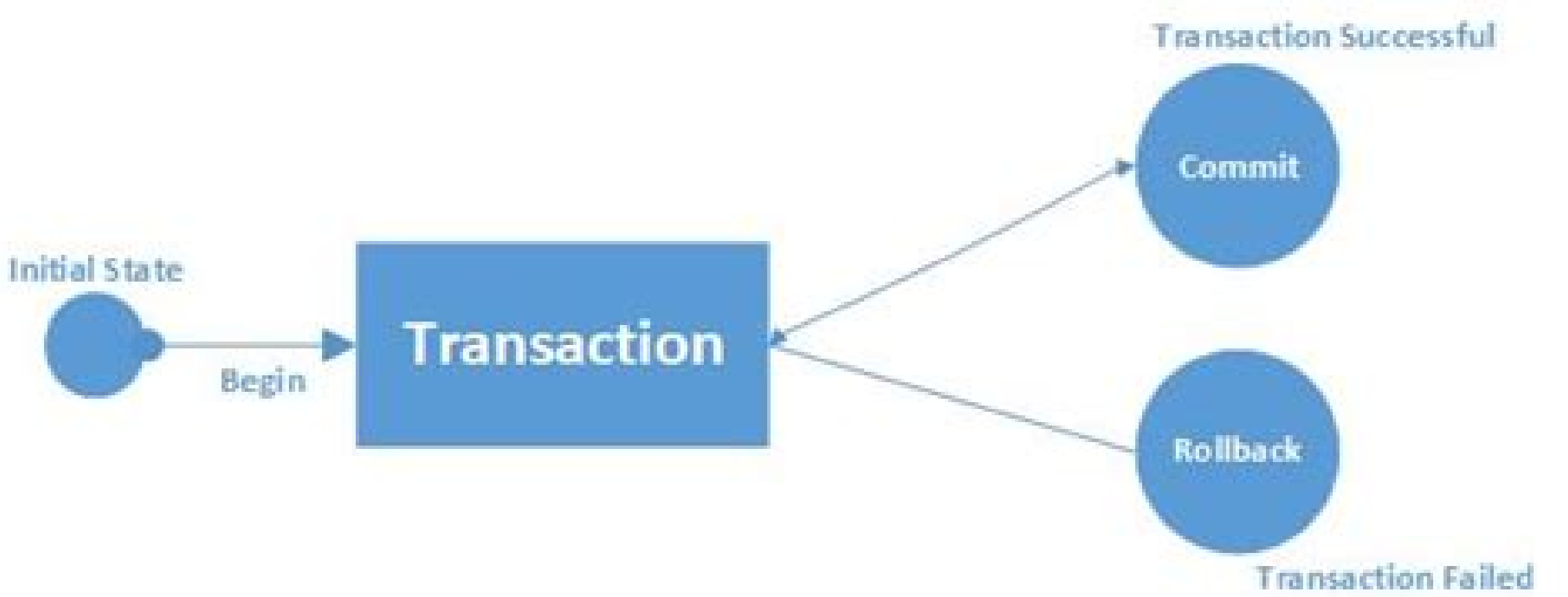
```
SELECT b.*
FROM country a
JOIN elevation b on b.id = a.id
WHERE a.name = 'France'
```

query.union()

```python
@app.route("/ressources_union/<string:nom>/<string:nom_>")
def ressources_union(nom, nom_):
    ressources = []

    query =  Country.query
    ressources_nom = query.\
        filter(Country.name == nom)
    ressources_nom_ = query.\
        filter(Country.name == nom_)

    ressources = ressources_nom.\
        union(ressources_nom_)._
        all()

    return render_template("pages/ressources.html",
        pays=nom, ressources=ressources,
        sous_titre=nom_)
```

```sql
SELECT *
FROM (SELECT *
        FROM country a
        INNER JOIN country_resources b on b.id = a.id
        INNER JOIN resources c on c.id = b.resource
        WHERE a.name = 'France'
    UNION
        SELECT *
        FROM country a
        INNER JOIN country_resources b on b.id = a.id
        INNER JOIN resources c on c.id = b.resource
        WHERE a.name = 'Germany'
    )
```

Ecriture des données



INSERT

```python
# routes/generales.py /add_country

from ..models.factbook import Country
from ..app import db

@app.route("/add_country/<string:pays>
    /<string:id>/<string:type>")
def add_country(pays, id, type):

    nouveau_pays = Country(id=id ,
        name=pays ,
        type=type)

    db.session.add(nouveau_pays)
    db.session.commit()
```

```sql
INSERT INTO country(id,
    Introduction,
    name,
    type)
VALUES ('pr',
    null ,
    'Paris',
    'dependency');

COMMIT;
```

Exercice

Créer une route

`/add_ressource/<string:pays>/<string:ressource>`

qui puisse ajouter une ressource à un pays

- identifier le pays sur lequel ajouter cette ressource
- identifier la ressource à ajouter
- ajouter la resource aux ressources du pays
- effectuer la transaction

Correction

```
# routes/generales.py /add_ressource
from ..models.factbook import Country
from ..app import db

@app.route("/add_ressource/<string:pays>/<string:ressource>")
def add_ressource(pays, ressource):
    pays = Country.\
        query.\
        filter(Country.name == pays).first()
    ressource = Resources.\
        query.\
        filter(Resources.name == ressource).first()
    pays.resources.append(ressource)
    db.session.add(pays)
    db.session.commit()
```

UPDATE

```python
# routes/generales.py /update_ressource

from ..models.factbook import  Resources

@app.route("/update_ressource/<string:current_name>
    /<string:new_name>")
def update_ressource(current_name, new_name):

    Resources.query.\
        filter(Resources.name == current_name).\
        update({"name": new_name})
    db.session.commit()
```

```sql
UPDATE resources
SET name = 'pétrole'
WHERE name = 'petroleum';

COMMIT;
```

DELETE

```
@app.route("/delete_ressource_by_query
    /<string:name>")
def delete_ressource_by_query(name):

    Resources.query.\
        filter(Resources.name == name).\
        delete()
    db.session.commit()
```

```
DELETE FROM resources
WHERE name = 'petroleum';

COMMIT;
```

Pagination des résultats

Bienvenue sur l'application du Factbook!

Voici le lien vers le site officiel : https://www.cia.gov/the-world-factbook/

1 2 ... 10 11 12 13 14 15 16 ... 26 27

#	Nom	Type	Description
1	Nicaragua	sovereign	The Pacific coast of Nicaragua was settled as a Spanish colony from Panama in the early 1 1821 and the country became an independent republic in 1838. Britain occupied the Carib gradually ceded control of the region in subsequent decades. Violent opposition to gover by 1978 and resulted in a short-lived civil war that brought a civic-military coalition, spear

Imiter SQL avec LIMIT et OFFSET

```
@app.route("/pagination_sql/<int:numero_page>
    /<int:nb_resultats>")
def pagination_sql(numero_page, nb_resultats):
    donnees = []

    query =  Country.query
    tous_resultats = query.\
        limit(nb_resultats * numero_page).\
        offset(numero_page).\
        all()
```

```
SELECT *
FROM country
LIMIT 20
OFFSET 2
```

La méthode `paginate()`

3 principaux paramètres :

- `page` : numéro de la page, début à 1
- `per_page` : nombre de résultats par page
- `error_out` : si `True`, une erreur 404 est retournée s'il n'y a plus de résultats

`paginate()` : configurations de l'application

```
# .env

...
PAYS_PER_PAGE=10
```

```
# config.py Config()

...
    PAYS_PER_PAGE = os.environ.get("PAYS_PER_PAGE")
```

`paginate()` : route

```
# routes/generales.py

...
@app.route("/pays/<int:page>")
def pays(page):
    donnees = []

    query =  Country.query
    tous_resultats = query.\
        paginate(page=page,
            per_page=app.config["PAYS_PER_PAGE"]
        )
```

paginate() : utilisation dans template

```
<!-- templates/pages/pays.html-->
{%if pagination%}
    <table class="table"><thead>
            <tr>
                <th scope="col">#</th>
                <th scope="col">Nom</th>
                <th scope="col">Type</th>
                <th scope="col">Description</th>
            </tr>
        </thead><tbody>
            {%for pays in pagination.items%}
            <tr>
                <th scope="row">{{loop.index}}</th>
                <td>{{pays.name}}</td>
                <td>{{pays.type}}</td>
                <td>{{pays.Introduction}}</td>
            </tr>
            {%endfor%}
        </tbody></table>
{%endif%}
```

`paginate()` : barre de navigation

```
<nav aria-label="research-pagination">
        <ul class="pagination">
            {%- for page in pagination.iter_pages() %}
                {% if page %}
                    {% if page != pagination.page %}
                        <li class="page-item">
                            <a class="page-link" href="{{ url_for("pays", page=page+1) }}">{{page}}</a>
                        </li>
                    {% else %}
                        <li class="page-item active disabled">
                        <a class="page-link">{{page}} <span class="sr-only">(actuelle)</span></a>
                        </li>
                    {% endif %}
                {% else %}
                    <li class="page-item disabled">
                    <a class="page-link">...</a>
                    </li>
                {% endif %}
            {%- endfor %}
        </ul>
    </nav>
```