Question: Explain how migrating from relational databases to object-relational databases allows eliminating the junction tables and related data directly in entity tables. **(04.00)**

Case study: Global Research Data Platform

SciData is a platform used by researchers worldwide to manage scientific experiments, publications, and datasets. Initially, it uses a ***relational SQL database***. The initial database is structured as following:

`Project``(project_id, title, start_date, end_date, field)`

`Researcher``(researcher_id, name, institution, field)`

`ResearcherProject``(researcher_id, project_id, role, contribution_percentage)`

`Paper``(paper_id, title, submission_date, publishing_date)`

`PaperResearcher``(paper_id, researcher_id, contribution_percentage)`

Part I – Legacy Relational SQL Database (5 points)

I.1. A researcher is allowed to participate **only in projects belonging to the same scientific field** as the researcher.
Write a **trigger function (without trigger)** that enforces this constraint during insertion or update operations on the ResearcherProject table.

I.2. Write an SQL **function** that returns the list of papers authored by a given researcher **for which their contribution percentage is the highest among all co-authors**.

Part II – Migration to Object-Relational Model (5 points)

II.1. Identify which tables from the initial relational schema can be **eliminated** when migrating to an **object-relational database**. Clearly justify your answer.

II.2. Describe, step by step, how to eliminate the `ResearcherProject` table **without losing legacy data or constraints**, taking into account:

- Existing many-to-many relationships
- Data migration

Hint: Your answer should include a conceptual object-relational schema using **user-defined types, nested collections, or composite attributes**.

Part III – distributed database (5 points)

III.1. Propose a **horizontal data fragmentation strategy** for the Project table based on the `field` attribute. Provide the SQL queries that implement this fragmentation.

III.2. Which **fragmentation strategy** can be applied to the `Researcher` table based on `researcher_id` in a distributed environment? Explain your choice conceptually, without writing SQL queries.

Part IV – NoSQL Database (5 points)

Assume that SciData adopts a **NoSQL document-oriented database** to store the same information.

IV.1. Provide an example of an **embedded JSON document** representing researchers, their projects, and their publications.

IV.2. Write a **NoSQL query** that returns the list of projects for a given researcher (id:101) **where the researcher acts as the project leader (principal investigator)**.

Dr.Belgroun Brahim

Question: Expliquez comment la migration d'une base de données relationnelle vers une base de données objet-relationnelle permet de **supprimer les tables de jonction** et d'intégrer directement les données associées dans les tables d'entité. **(04.00)**

Étude de cas: Plateforme mondiale de données scientifiques

SciData est une plateforme utilisée par des chercheurs du monde entier pour gérer des expériences scientifiques, des publications et des ensembles de données. La base de données initiale est relationnelle et structurée comme suit :

Project(project_id, title, start_date, end_date, field)

Researcher(researcher_id, name, institution, field)

ResearcherProject(researcher_id, project_id, role, contribution_percentage)

Paper(paper_id, title, submission_date, publishing_date)

PaperResearcher(paper_id, researcher_id, contribution_percentage)

Partie I – Base de données relationnelle classique (5 points)

I.1. Un chercheur ne peut participer qu'aux projets appartenant au **même domaine scientifique** que lui. Écrivez une **fonction trigger** (sans créer le trigger) qui applique cette contrainte lors d'une insertion ou d'une mise à jour dans la table ResearcherProject.

I.2. Écrivez une **fonction SQL** qui retourne la liste des articles rédigés par un chercheur donné pour lesquels **son pourcentage de contribution est le plus élevé parmi tous les co-auteurs**.

Partie II – Migration vers le model objet-relationnelle (5 points)

II.1. Identifiez quelles tables du schéma relationnel initial peuvent être **supprimées** lors de la migration vers une base objet-relationnelle. Justifiez clairement votre réponse.

II.2. Décrivez, **étape par étape**, comment supprimer la table ResearcherProject sans perdre les données anciennes, en prenant en compte :

- Les relations plusieurs-à-plusieurs existantes
- La migration des données

Indice : Votre réponse doit inclure un schéma conceptuel objet-relationnel utilisant des types définis par l'utilisateur, des collections imbriquées ou des attributs composites.

Partie III – Base de données distribuée (5 points)

III.1. Proposez une **stratégie de fragmentation horizontale** pour la table Project basée sur l'attribut "field". Fournissez les **requêtes SQL** permettant de réaliser cette fragmentation.

III.2. Quelle **stratégie de fragmentation** peut être appliquée à la table ***Researcher*** basée sur ***researcher_id*** dans un environnement distribué ? Expliquez votre choix **conceptuellement**, sans écrire de requêtes SQL.

Partie IV – Base de données NoSQL (5 points)

Assume that SciData adopts a **NoSQL document-oriented database** to store the same information.

IV.1. Donnez un exemple de **document JSON imbriqué** représentant les chercheurs, leurs projets et leurs publications.

IV.2. Écrivez une **requête NoSQL** qui retourne la liste des projets pour un chercheur donné (id:101) où le chercheur est **le responsable principal du projet**.

Dr.Belgroun Brahim

Solution:

Question: Migrating from relational databases to object-relational databases allows eliminating junction tables because ORDBs support **nested collections, composite types, and direct object references**. In relational databases, many-to-many relationships require a separate table to store foreign keys and any relationship attributes. In an ORDB, these relationships can be represented **directly inside the entity tables**, embedding the related entities and their attributes as collections or objects. **(04.00)**

Case study: Global Research Data Platform

Part I – Legacy Relational SQL Database (5 points)

I.1. The trigger function

```
CREATE OR REPLACE FUNCTION
check_researcher_project_field()
RETURNS TRIGGER AS $$
DECLARE
    researcher_field TEXT;
    project_field TEXT;
BEGIN
    SELECT field INTO researcher_field
    FROM Researcher
    WHERE researcher_id = NEW.researcher_id;
    SELECT field INTO project_field
    FROM Project
    WHERE project_id = NEW.project_id;
    IF researcher_field <> project_field THEN
        RAISE EXCEPTION
        'Researcher field (%) does not match project
field (%)',
        researcher_field, project_field;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

I.2. SQL Function

```
CREATE OR REPLACE FUNCTION
papers_with_max_contribution(p_researcher_id INT)
RETURNS TABLE (
    paper_id INT,
    title TEXT,
    contribution_percentage NUMERIC
) AS $$
BEGIN
    RETURN QUERY
    SELECT p.paper_id, p.title,
pr.contribution_percentage
    FROM Paper p
    JOIN PaperResearcher pr
        ON p.paper_id = pr.paper_id
    WHERE pr.researcher_id = p_researcher_id
      AND pr.contribution_percentage = (
          SELECT MAX(pr2.contribution_percentage)
          FROM PaperResearcher pr2
          WHERE pr2.paper_id = p.paper_id
      );
END;
$$ LANGUAGE plpgsql;
```

Part II – Migration to Object-Relational Database (5 points)

II.1. Tables that can be eliminated

The two tables `ResearcherProject` and `PaperResearcher` can be eliminated car they **only exist** to represent **many-to-many relationships.** So their attributes can be **embedded** using Nested collections, Composite types, and/or Object references.

II.2. Step-by-step elimination of `ResearcherProject`

Step 1 – Define object types

```
CREATE TYPE ProjectParticipation AS (
    project_id INT,
    role TEXT,
    contribution_percentage NUMERIC
);
```

Step 2 – Extend Researcher table

```
ALTER TABLE Researcher
ADD COLUMN projects ProjectParticipation[];
```

Step 3 – Migrate legacy data

```
UPDATE Researcher r
```

```
SET projects = (
    SELECT ARRAY_AGG(
        (rp.project_id,
        rp.role,
        rp.contribution_percentage
    )::ProjectParticipation)
    FROM ResearcherProject rp
    WHERE rp.researcher_id = r.researcher_id
);
```

Step 4 – Drop junction table

```
DROP TABLE ResearcherProject;
```

Part III – distributed database (5 points)

III.1. Horizontal fragmentation of Project by field

```
CREATE TABLE Project (
    project_id SERIAL PRIMARY KEY,
    title TEXT NOT NULL,
    start_date DATE,
    end_date DATE,
    field TEXT NOT NULL
) PARTITION BY LIST (field);

CREATE TABLE Project_Biology PARTITION OF Project
FOR VALUES IN ('Biology');

CREATE TABLE Project_ComputerScience PARTITION OF Project
FOR VALUES IN ('Computer Science');

CREATE TABLE Project_Physics PARTITION OF Project
FOR VALUES IN ('Physics');

CREATE TABLE Project_Other PARTITION OF Project
DEFAULT;
```

III.2. horizontal fragmentation strategy

For the Researcher table in a distributed environment, a **hash-based horizontal fragmentation strategy** can be applied using researcher_id as the key.

Hash-based horizontal fragmentation divides the table into multiple fragments by applying a **hash function** to the researcher_id. Each resulting hash value determines the fragment (or server) where the researcher's row is stored.

Part IV – NoSQL Database (5 points)

IV.1. Example of an **embedded JSON document**

```
{  "researcher_id": 101,
 "name": "Dr. Alice Martin",
 "institution": "Global Science Lab",
 "field": "Computer Science",
 "projects": [
  {
   "project_id": 501,
   "title": "AI for Healthcare",
   "role": "Principal Investigator",
   "contribution_percentage": 60,
   "papers": [
    {
     "paper_id": 9001,
     "title": "Deep Learning Models",
     "contribution_percentage": 70
    }
]
}
]
}
```

IV.2. **NoSQL query**

```
db.researchers.find(
  {
    researcher_id: 101,
    "projects.role": "Leader"
  },
  {
    "projects.$": 1,
    name: 1
  }
);
```

Dr.Belgroun Brahim