Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**«Национальный исследовательский ядерный университет «МИФИ»**

**ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ**

**Кафедра №42 (криптологии и кибербезопасности)**

ЛАБОРАТОРНАЯ РАБОТА №1-3  
по дисциплине: «Безопасность систем баз данных»  
на тему: «Сложные запросы на выборку. Соединения»

**Выполнил:**студент группы Б19-515  
Родионов Дмитрий Александрович

**Москва, 2022**

Оглавление

[Описание хода работы 3](#_Toc102769320)

[Список выполненных сложных запросов SQL 5](#_Toc102769321)

[Приложения 9](#_Toc102769322)

[Заключение 16](#_Toc102769323)

# Описание хода работы

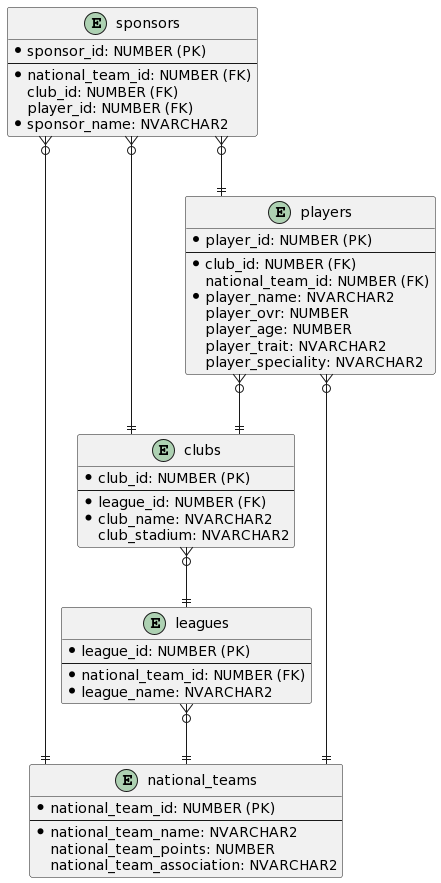
**1. Подготовка «легенды» – вопросов, на которые требуется знать ответы среднестатистическим пользователям игры.**

**2. Разработка и проверка следующих типов SQL-запросов, реализующих данные ответы:**

* подзапросы,
* соединение,
* иерархические запросы,
* аналитические функции,
* теоретико-множественные операции.

**3. Разработка 8 различных запросов с применением описанных возможностей, предоставляемых стандартом Oracle Database.**

**Диаграмма отношения сущностей:**



# Список выполненных сложных запросов SQL

**1. Подзапросы: вместо переменной или литерала в запросе можно использовать вложенный подзапрос.** Этот подзапрос должен возвращать либо одно значение, либо столбец значений, если оператор работает с диапазонами. Например, этот запрос вернет список игроков, которые представляют футбольные ассоциации UEFA:

SELECT \* FROM players WHERE national\_team\_id IN (SELECT national\_team\_id FROM national\_teams WHERE national\_team\_association = 'UEFA');

Этот запрос вернет обновленные рейтинги игроков с учетом выступления представляемых ими национальных футбольных сборных:

SELECT player\_name, (

2 SELECT ROUND(player\_ovr \* (

3 SELECT national\_team\_points / (

4 SELECT AVG(national\_team\_points) AS national\_team\_coefficient

5 FROM national\_teams) AS national\_team\_coefficient

6 FROM national\_teams

7 WHERE national\_team\_id = players.national\_team\_id))

8 FROM dual) AS player\_ovr\_upgrade

9 FROM players

10 ORDER BY player\_ovr\_upgrade DESC;

**2. Соединение.** Такой запрос сопоставляет друг другу ряды из двух таблиц. Без дополнительной условной фильтрации каждая строка из одной таблицы соединяется с каждой строкой из другой. Операция соединения применяется для денормализации данных. Например, этот запрос позволяет извлечь список игроков с указанием футбольного клуба, в котором они выступают:

SELECT clubs.club\_name, players.player\_name FROM clubs, players WHERE clubs.club\_id = players.club\_id;

Таблицу можно соединять саму с собой. Например, этот запрос выводит список потенциальных пар команд, которые могут встретиться друг с другом на континентальном турнире:

SELECT national\_team1.national\_team\_name, national\_team2.national\_team\_name, national\_team1.national\_team\_association

2 FROM national\_teams national\_team1, national\_teams national\_team2

3 WHERE national\_team1.national\_team\_name != national\_team2.national\_team\_name

4 AND national\_team1.national\_team\_association = national\_team2.national\_team\_association;

**3. Иерархические запросы (CONNECT BY, PRIOR, START WITH).** Строятся к таблицам, где одни записи подчинены другим. Типичные примеры: таблица сотрудников, где для каждого указан непосредственный начальник; таблица обработки материалов, где для каждого типа сырья указан производимый из него продукт. При выполнении иерархического запроса создаются дополнительные столбцы LEVEL, CONNECT\_BY\_ISCYCLE, CONNECT\_BY\_ISLEAF, а также доступна функция CONNECT\_BY\_PATH. Например, этот запрос выводит уровень футбольного клуба (лиги) в иерархии:

SELECT club\_id,  
 2 league\_id,  
 3 club\_name,  
 4 RPAD('.', (LEVEL - 1) \* 2, '.') || club\_id AS tree,  
 5 LEVEL,  
 6 CONNECT\_BY\_ROOT club\_id AS root\_id,  
 7 LTRIM(SYS\_CONNECT\_BY\_PATH(club\_id, '-'), '-') AS path,  
 8 CONNECT\_BY\_ISLEAF AS leaf,  
 9 CONNECT\_BY\_IS CYCLE AS cycle  
 10 FROM clubs  
 11 START WITH club\_id = 1  
 12 CONNECT BY NOCYCLE PRIOR club\_id = league\_id  
 13 ORDER SIBLINGS BY club\_id;

**4. Аналитические функции (OVER).** Предназначены для анализа того, как значения полей меняются с течением времени (или любого другого параметра). Позволяют извлечь значение заданного поля не для текущей записи, а для предшествующей; вычислить скользящее среднее и так далее. Так как аналитические функции оперируют понятиями «предыдущий», «следующий», «первый», – их применение осмысленно только при наличии сортировки. Поэтому параметр и порядок сортировки при использовании аналитической функции нужно обязательно указывать. При этом, результаты запроса можно отсортировать по-другому. Пример: этот запрос выводит рейтинг ближайшей по уровню команды в рамках футбольной ассоциации:

SELECT national\_team\_name,  
 2 national\_team\_points,  
 3 LAG(national\_team\_points, 1, NULL)  
 4 OVER(ORDER BY national\_team\_points)  
 5 AS previous\_national\_team\_points  
 6 FROM national\_teams;

Этот запрос вычисляет рейтинг футбольной ассоциации с учетом уровня входящих в нее национальных сборных:

SELECT national\_team\_association,  
 2 MAX(national\_team\_association\_points)  
 3 AS max\_national\_team\_association\_points  
 4 FROM (  
 5 SELECT national\_team\_name,  
 6 national\_team\_association,  
 7 SUM(national\_team\_points)  
 8 OVER(  
 9 PARTITION BY national\_team\_association  
 10 ORDER BY national\_team\_points  
 11 ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)  
 12 AS national\_team\_association\_points  
 13 FROM national\_teams)  
 14 GROUP BY national\_team\_association;

**5. Теоретико-множественные операции (UNION, UNION ALL, INTERSECT, MINUS).** Позволяют построить объединение, пересечение, разность множеств из двух таблиц. Ценность этих инструкций состоит в возможности объединить две различных таблицы в одну. Оператор UNION устраняет возникающие при объединении дубликаты, UNION ALL этого не делает, поэтому работает существенно быстрее. Например, этот запрос возвращает список национальных сборных и компаний, выступающих в роли спонсоров этих национальных сборных:

SELECT national\_team\_id,  
 2 national\_team\_name,  
 3 CAST('' AS NVARCHAR2(50)) AS sponsor\_name  
 4 FROM national\_teams  
 5 UNION  
 6 SELECT national\_team\_id,  
 7 CAST('' AS NVARCHAR2(60)) AS national\_team\_name,  
 8 sponsor\_name  
 9 FROM sponsors;

# Приложения

**Листинг использованных инструкций SQL:**

SQL> SELECT \* FROM players WHERE national\_team\_id IN (SELECT national\_team\_id FROM national\_teams WHERE national\_team\_points > 1700);

SQL> SELECT \* FROM players WHERE national\_team\_id IN (SELECT national\_team\_id FROM national\_teams WHERE national\_team\_association = 'UEFA');

SQL> SELECT \* FROM sponsors WHERE player\_id IN (SELECT player\_id FROM players WHERE player\_ovr > 80);

SQL> SELECT player\_name, (

2 SELECT ROUND(player\_ovr \* (

3 SELECT national\_team\_points / (

4 SELECT AVG(national\_team\_points) AS national\_team\_coefficient

5 FROM national\_teams) AS national\_team\_coefficient

6 FROM national\_teams

7 WHERE national\_team\_id = players.national\_team\_id))

8 FROM dual) AS player\_ovr\_upgrade

9 FROM players

10 ORDER BY player\_ovr\_upgrade DESC;

SQL> SELECT national\_teams.national\_team\_name, leagues.league\_name FROM national\_teams, leagues WHERE national\_teams.national\_team\_id = leagues.national\_team\_id;

SQL> SELECT national\_teams.national\_team\_name, players.player\_name FROM national\_teams, players WHERE national\_teams.national\_team\_id = players.national\_team\_id;

SQL> SELECT leagues.league\_name, clubs.club\_name FROM leagues, clubs WHERE leagues.league\_id = clubs.league\_id;

SQL> SELECT clubs.club\_name, players.player\_name FROM clubs, players WHERE clubs.club\_id = players.club\_id;

SQL> SELECT national\_team1.national\_team\_name, national\_team2.national\_team\_name, national\_team1.national\_team\_association

2 FROM national\_teams national\_team1, national\_teams national\_team2

3 WHERE national\_team1.national\_team\_name != national\_team2.national\_team\_name

4 AND national\_team1.national\_team\_association = national\_team2.national\_team\_association;

SQL> SELECT league1.league\_name, (

2 league2.league\_name, (

3 SELECT national\_team\_name FROM national\_teams WHERE national\_team\_id = league1.national\_team\_id) AS national\_team\_name

4 FROM leagues league1, leagues league2

5 WHERE league1.league\_name != league2.league\_name

6 AND league1.national\_team\_id = league2.national\_team\_id;

SQL> SELECT club1.club\_name,

2 club2.club\_name, (

3 SELECT league\_name FROM leagues WHERE league\_id = club1.league\_id) AS league\_name

4 FROM clubs club1, clubs club2

5 WHERE club1.club\_id != club2.club\_id

6 AND club1.league\_id = club2.league\_id;

SQL> SELECT player1.player\_name, (

2 player2.player\_name, (

3 SELECT club\_name FROM clubs WHERE club\_id = player1.club\_id) AS club\_name

4 FROM players player1, players player2

5 WHERE player1.player\_id != player2.player\_id

6 AND player1.club\_id = player2.club\_id;

SQL> SELECT player1.player\_name, (

2 player2.player\_name, (

3 SELECT national\_team\_name FROM national\_teams WHERE national\_team\_id = player1.national\_team\_id) AS national\_team\_name

4 FROM players player1, players player2

5 WHERE player1.player\_id != player2.player\_id

6 AND player1.national\_team\_id = player2.national\_team\_id;

SQL> SELECT league\_id,

2 national\_team\_id,

3 league\_name,

4 RPAD('.', (LEVEL - 1) \* 2, '.') || league\_id AS tree,

5 LEVEL,

6 CONNECT\_BY\_ROOT league\_id AS root\_id,

7 LTRIM(SYS\_CONNECT\_BY\_PATH(league\_id, '-'), '-') AS path,

8 CONNECT\_BY\_ISLEAF AS leaf,

9 CONNECT\_BY\_ISCYCLE AS cycle

10 FROM leagues

11 START WITH league\_id = 1

12 CONNECT BY NOCYCLE PRIOR league\_id = national\_team\_id

13 ORDER SIBLINGS BY league\_id;

SQL> SELECT club\_id,

2 league\_id,

3 club\_name,

4 RPAD('.', (LEVEL - 1) \* 2, '.') || club\_id AS tree,

5 LEVEL,

6 CONNECT\_BY\_ROOT club\_id AS root\_id,

7 LTRIM(SYS\_CONNECT\_BY\_PATH(club\_id, '-'), '-') AS path,

8 CONNECT\_BY\_ISLEAF AS leaf,

9 CONNECT\_BY\_ISCYCLE AS cycle

10 FROM clubs

11 START WITH club\_id = 1

12 CONNECT BY NOCYCLE PRIOR club\_id = league\_id

13 ORDER SIBLINGS BY club\_id;

SQL> SELECT national\_team\_name,

2 national\_team\_points,

3 MAX(national\_team\_points)

4 OVER()

5 AS max\_national\_team\_points

6 FROM national\_teams

7 ORDER BY national\_team\_points DESC;

SQL> SELECT national\_team\_name,

2 national\_team\_points,

3 MAX(national\_team\_points)

4 OVER(PARTITION BY national\_team\_association)

5 AS max\_national\_team\_points

6 FROM national\_teams

7 ORDER BY national\_team\_points DESC;

SQL> SELECT player\_name,

2 player\_ovr,

3 player\_age,

4 TRUNC(AVG(player\_age))

5 OVER()

6 AS avg\_player\_age

7 FROM players

8 ORDER BY player\_ovr DESC;

SQL> SELECT player\_name,

2 club\_id,

3 player\_ovr,

4 player\_age,

5 AVG(player\_age)

6 OVER(PARTITION BY club\_id)

7 AS avg\_player\_age

8 FROM players

9 ORDER BY player\_ovr DESC;

SQL> SELECT player\_name,

2 player\_ovr,

3 player\_age,

4 MIN(player\_ovr)

5 OVER()

6 AS min\_player\_ovr

7 FROM players

8 ORDER BY player\_ovr DESC;

SQL> SELECT player\_name,

2 national\_team\_id,

3 player\_ovr,

4 player\_age,

5 MIN(player\_ovr)

6 OVER(PARTITION BY national\_team\_id)

7 AS min\_player\_ovr

8 FROM players

9 ORDER BY player\_ovr DESC;

SQL> SELECT national\_team\_name,

2 national\_team\_points,

3 LAG(national\_team\_points, 1, NULL)

4 OVER(ORDER BY national\_team\_points)

5 AS previous\_national\_team\_points

6 FROM national\_teams;

SQL> SELECT national\_team\_name,

2 national\_team\_points,

3 LAG(national\_team\_points, 1, NULL)

4 OVER(

5 PARTITION BY national\_team\_association

6 ORDER BY national\_team\_points DESC)

7 AS next\_national\_team\_points

8 FROM national\_teams;

SQL> SELECT player\_name,

2 club\_id,

3 player\_ovr,

4 player\_trait,

5 LAG(player\_trait, 1, NULL)

6 OVER(

7 PARTITION BY club\_id

8 ORDER BY player\_ovr DESC)

9 AS next\_player\_trait

10 FROM players;

SQL> SELECT player\_name,

2 national\_team\_id,

3 player\_ovr,

4 player\_speciality,

5 LAG(player\_speciality, 1, NULL)

6 OVER(

7 PARTITION BY national\_team\_id

8 ORDER BY player\_ovr DESC)

9 AS next\_player\_speciality

10 FROM players;

SQL> SELECT national\_team\_name,

2 national\_team\_association,

3 SUM(national\_team\_points)

4 OVER(

5 PARTITION BY national\_team\_association

6 ORDER BY national\_team\_points

7 ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

8 AS national\_team\_association\_points

9 FROM national\_teams;

SQL> SELECT national\_team\_association,

2 MAX(national\_team\_association\_points)

3 AS max\_national\_team\_association\_points

4 FROM (

5 SELECT national\_team\_name,

6 national\_team\_association,

7 SUM(national\_team\_points)

8 OVER(

9 PARTITION BY national\_team\_association

10 ORDER BY national\_team\_points

11 ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

12 AS national\_team\_association\_points

13 FROM national\_teams)

14 GROUP BY national\_team\_association;

SQL> SELECT player\_name,

2 club\_id,

3 TRUNC(AVG(player\_ovr))

4 OVER(

5 PARTITION BY club\_id

6 ORDER BY player\_ovr

7 ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

8 AS avg\_player\_club\_ovr

9 FROM players

10 ORDER BY club\_id;

SQL> SELECT national\_team\_id,

2 TRUNC(AVG(player\_national\_team\_ovr))

3 AS avg\_player\_national\_team\_ovr

4 FROM (

5 SELECT player\_name,

6 national\_team\_id,

7 AVG(player\_ovr)

8 OVER(

9 PARTITION BY national\_team\_id

10 ORDER BY player\_ovr

11 ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)

12 AS player\_national\_team\_ovr

13 FROM players)

14 GROUP BY national\_team\_id;

SQL> SELECT 3 FROM dual

2 INTERSECT

3 SELECT 3f FROM dual;

SQL> SELECT national\_team\_id,

2 national\_team\_name,

3 CAST('' AS NVARCHAR2(50)) AS league\_name

4 FROM national\_teams

5 UNION

6 SELECT national\_team\_id,

7 CAST('' AS NVARCHAR2(60)) AS national\_team\_name,

8 league\_name

9 FROM leagues;

SQL> SELECT national\_team\_id,

2 national\_team\_name,

3 CAST('' AS NVARCHAR2(50)) AS player\_name

4 FROM national\_teams

5 UNION

6 SELECT national\_team\_id,

7 CAST('' AS NVARCHAR2(60)) AS national\_team\_name,

8 player\_name

9 FROM players;

SQL> SELECT national\_team\_id,

2 national\_team\_name,

3 CAST('' AS NVARCHAR2(50)) AS sponsor\_name

4 FROM national\_teams

5 UNION

6 SELECT national\_team\_id,

7 CAST('' AS NVARCHAR2(60)) AS national\_team\_name,

8 sponsor\_name

9 FROM sponsors;

SQL> SELECT league\_id,

2 league\_name,

3 CAST('' AS NVARCHAR2(50)) AS club\_name

4 FROM leagues

5 UNION

6 SELECT league\_id,

7 CAST('' AS NVARCHAR2(50)) AS league\_name,

8 club\_name

9 FROM clubs;

SQL> SELECT club\_id,

2 club\_name,

3 CAST('' AS NVARCHAR2(50)) AS player\_name

4 FROM clubs

5 UNION ALL

6 SELECT club\_id,

7 CAST('' AS NVARCHAR2(50)) AS club\_name,

8 player\_name

9 FROM players

10 ORDER BY club\_id;

SQL> SELECT club\_id,

2 club\_name,

3 CAST('' AS NVARCHAR2(50)) AS sponsor\_name

4 FROM clubs

5 UNION ALL

6 SELECT club\_id,

7 CAST('' AS NVARCHAR2(50)) AS club\_name,

8 sponsor\_name

9 FROM sponsors

10 ORDER BY club\_id;

SQL> SELECT player\_id,

2 player\_name,

3 CAST('' AS NVARCHAR2(50)) AS sponsor\_name

4 FROM players

5 UNION ALL

6 SELECT player\_id,

7 CAST('' AS NVARCHAR2(50)) AS player\_name,

8 sponsor\_name

9 FROM sponsors

10 ORDER BY player\_id;

SQL> SELECT national\_team\_id,

2 national\_team\_name,

3 CAST('' AS NVARCHAR2(50)) AS league\_name

4 FROM national\_teams

5 INTERSECT

6 SELECT national\_team\_id,

7 CAST('' AS NVARCHAR2(60)) AS national\_team\_name,

8 league\_name

9 FROM leagues;

SQL> SELECT national\_team\_id,

2 national\_team\_name,

3 CAST('' AS NVARCHAR2(50)) AS league\_name

4 FROM national\_teams

5 MINUS

6 SELECT national\_team\_id,

7 CAST('' AS NVARCHAR2(60)) AS national\_team\_name,

8 league\_name

9 FROM leagues;

SQL> SELECT national\_team\_id,

2 CAST('' AS NVARCHAR2(60)) AS national\_team\_name,

3 league\_name

4 FROM leagues

5 MINUS

6 SELECT national\_team\_id,

7 national\_team\_name,

8 CAST('' AS NVARCHAR2(50)) AS league\_name

9 FROM national\_teams;

SQL> SELECT national\_teams.national\_team\_id,

2 national\_teams.national\_team\_name,

3 sponsors.sponsor\_name

4 FROM national\_teams

5 LEFT OUTER JOIN sponsors

6 ON national\_teams.national\_team\_id = sponsors.national\_team\_id;

SQL> SELECT players.club\_id,

2 players.player\_name,

3 clubs.club\_name

4 FROM clubs

5 RIGHT OUTER JOIN players

6 ON clubs.club\_id = players.club\_id;

SQL> SELECT

2 national\_teams.national\_team\_name,

3 leagues.league\_name,

4 clubs.club\_name,

5 players.player\_name,

6 players.player\_ovr

7 FROM

8 national\_teams,

9 leagues,

10 clubs,

11 players

12 WHERE

13 national\_teams.national\_team\_id = leagues.national\_team\_id

14 AND

15 leagues.league\_id = clubs.league\_id

16 AND

17 clubs.club\_id = players.club\_id

18 ORDER BY players.player\_ovr DESC;

SQL> WITH

2 summary AS (

3 SELECT national\_teams.national\_team\_name, leagues.league\_name, clubs.club\_name, players.player\_name, players.player\_ovr

4 FROM national\_teams, leagues, clubs, players

5 WHERE national\_teams.national\_team\_id = leagues.national\_team\_id AND leagues.league\_id = clubs.league\_id AND clubs.club\_id = players.club\_id

6 ORDER BY players.player\_ovr DESC),

7 helper AS (

8 SELECT DISTINCT NULL leaf, NULL child, NULL parent, national\_team\_name root FROM summary WHERE national\_team\_name = 'Belgium'

9 UNION ALL

10 SELECT DISTINCT NULL leaf, NULL child, national\_team\_name parent, league\_name root FROM summary

11 UNION ALL

12 SELECT DISTINCT NULL leaf, national\_team\_name child, league\_name parent, club\_name root FROM summary

13 UNION ALL

14 SELECT DISTINCT national\_team\_name leaf, league\_name child, club\_name parent, player\_name root FROM summary),

15 cte AS (

16 SELECT leaf, child, parent, root,

17 RPAD(' ', (LEVEL - 1) \* 2, ' ') || root AS tree,

18 LEVEL,

19 CONNECT\_BY\_ROOT root,

20 LTRIM(SYS\_CONNECT\_BY\_PATH(root, ' | '), ' | ') AS path

21 FROM helper

22 CONNECT BY PRIOR root = parent)

23 SELECT \* FROM cte;  
  
SQL> EXIT

# Заключение

В данной лабораторной работе приобретены навыки написания сложных запросов на выборку к базе данных Oracle, в частности – с применением аналитических функций и теоретико-множественных операций. В ходе выполнения работы особое внимание уделено различным случаям соединения таблиц.

Цель, поставленная в начале работы, достигнута; задачи выполнены.