Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра информатики

# Пояснительная записка

к курсовому проекту

на тему

**Реализация АРХИТЕКТУРЫ БД ДЛЯ СЕРВИСА ПО ПРОГНОЗУ СПОРТИВНЫХ ДАННЫХ.**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Студент |  | А. П. Харкевич |
| Руководитель |  | В. С. Плиска |

Минск 2022

Министерство образования Республики Беларусь

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Учреждение образования БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ | | | | | | | | | |
|  | | | | | | | | | |
| Факультет | КС и С | | | | Кафедра | | Информатики | | |
| Специальность | 1-40 04 01 | | | | Специализация | | |  | |
| ЗАДАНИЕ | | | | | | | | | |
| по курсовому проекту студента | | | | | | | | | |
| Харкевича Антона Павловича | | | | | | | | | |
| (фамилия, имя, отчество) | | | | | | | | | |
| 1. Тема проекта: | | Реализация архитектуры БД для сервиса по прогнозу спортивных | | | | | | | |
| **данных** | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
| 2. Срок сдачи студентом законченной работы | | | | | |  | | | |
| 3. Исходные данные к проекту | | | | Тип операционной системы – ОС Windows; | | | | | |
| Языки программирования – SQL, PL/SQL; | | | | | | | | | |
|  | | | | | | | | | |
| Цель проекта: реализация архитектуры БД которая может быть интегрирована в сервис по прогнозу спортивных данных | | | | | | | | | |
|  | | | | | | | | | |
|  | | | | | | | | | |
| 4. Содержание пояснительной записки (перечень подлежащих разработке вопросов) | | | | | | | | | |
| Введение | | | | | | | | | |
| 1 Формирование требований к проекту | | | | | | | | | |
| 2 Основная идея реализации требований | | | | | | | | | |
| 3 Проектирование проекта | | | | | | | | | |
| 4 Программная реализация | | | | | | | | | |
| 5 Тестирование проекта | | | | | | | | | |
| Заключение | | | | | | | | | |
| Список использованных источников | | | | | | | | | |
| Приложение А - Текст программы | | | | | | | | | |
| 5. Перечень графического материала (с точным указанием наименования) и обозначения | | | | | | | | |
| вида и типа материала) | | |  | | | | | |
| Общая схема архитектуры проекта | | | | | | | | |
| Схема устройства кэша базы данных | | | | | | | | |
|  | | | | | | | | |
|  | | | | | | | | |
|  | | | | | | | | |

# КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование этапов дипломного проекта (работы) | Объём этапа в % | Срок выполнения этапа | Примечание |
| Анализ предметной |  |  |  |
| области, разработка технического задания | 15-20 | 11.09–15.10 |  |
| Разработка функциональных требований, |  |  |  |
| проектирование архитектуры программы | 20-15 | 16.10–15.11 |  |
| Разработка схемы программы, алгоритмов, |  |  |  |
| схемы данных | 20-15 | 16.11–15.12 |  |
| Разработка программного средства | 15-20 | 16.12– (в зависимости от курса) |  |
| Тестирование и отладка | 10 | 5 дней до сдачи |  |
| Оформление пояснительной записки |  |  |  |
| и графического материала | 20 | 2 дня до сдачи |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Дата выдачи задания |  | | Руководитель | |  | В.С. Плиска |
| Задание принял к исполнению | |  | | А.П. Харкевич | |

**ВВЕДЕНИЕ**

В наше время огромное количество фирм используют персональные компьютеры для сохранения и обработки любого вида информации. Эта информация содержится в базах данных. Базы данных играют важную роль в развивающемся мире технологий. Всё, с чем мы каждый день взаимодействуем в жизни, по всей видимости, зафиксировано в какой-нибудь базе. Работа с базами данных является важнейшим навыком в работе с компьютером, а специалисты данной области становятся всё более востребованными. Главные идеи нынешней информационной методики базируются на представлении, в соответствии чему информация должна быть образована в базы данных с задачей отображения динамически изменяющегося мира и удовлетворения всех потребностей в информации у пользователей. Базы данных формируются и работают под управлением специальных программных средств, называемых системами управления базами данных.

База данных, которая представлена в объективной форме, это совокупность таких материалов: статей, счетов, нормативных актов, судебных решений или иных подобных материалов, собранные вместе таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины.

База данных — это организованная структура, которая предназначена для хранения информации. В то время, когда происходило развитие термина баз данных, в них сохранялись исключительно информация, однако уже в наши дни многие системы управления базами данных позволяет размещать в своих структурах и данные, и программный код, с помощью которого совершается связь с пользователями или с другими программно-аппаратными комплексами. При этом данные должны не противоречить друг другу, целостны и не избыточными. База данных создается для сохранения и непосредственного доступа к информации, содержащей сведения об искомой предметной области. Степень конкретизации данных обуславливается группой факторов. Прежде всего, целью использования информации из баз данных и сложностью информационных процессов, существующих в пределах предметной области в конкретных условиях.

Система управления базами данных — это программный механизм, предназначенный для записи, поиска, обработки и печати информации, содержащейся в базе данных.

В компьютере данные базы данных представляется в виде таблицы, схожей на электронную таблицу. Названия столбцов, представляющих заголовки таблицы, называют именами полей, а сами столбцы - полями. Данные, которые находятся в полях, называют значениями полей.

Сами базы данных — это хранилища огромного множества систематизированной информации, с которыми производятся следующие операции: изменение, копирование, удаление, добавление. Накопление хранимого объема информации, рост группы пользователей информационных систем служат источником к обширному развитию самых комфортных в интерфейсе и относительно лёгких для понимания табличных систем управления базами данных. Создание доступа к информации базы данных сразу нескольких пользователей одновременного, зачастую находящихся на далеком расстоянии от места хранения баз данных, а также друг от друга, поэтому и созданы многопользовательские сетевые версии баз данных, сформированные на табличной структуре. В них решаются проблемы характерные для параллельных процессов, правильности данных, а также получения не санкционированного входа.

Любая современная организация не может обойтись без базы данных. Это учебные заведения, банки, магазины, заводы, любые предприятия и государственные учреждения. Они используют их для перевода данных в электронный вид и объединения данных, а также оперативного доступа к ним. Это позволяет экономить время и средства на затраты.

Конечно, снижение времени является лишь побочным эффектом автоматизации. Самая главная задача развития информационных технологий в совсем другом - в приобретении той или иной организацией исключительно новых качеств, придающих ей существенную конкурентоспособность. А это дорогого стоит.

**Формирование требований к проекту**

Целью проекта является реализация архитектуры БД которая может быть интегрирована в сервис по прогнозу спортивных данных.

Далее следует краткое описание сервиса по прогнозу спортивных матчей.

Пользователем приложения по анализу подписок будет являться любой человек зашедший на сайт, авторизованным пользователем будет являться человек, прошедший регистрацию в приложении, вовлеченным пользователем будет считать человек, прошедший регистрацию и купивший хоть одну подписку.

Пользователями сервиса по анализу прибыли будут являться заказчик или его представители.

Подписка дает возможность пользователю узнавать прогноз результата интересующего его матча.

Существуют подписки следующих типов:

пробная – действует 7 дней и позволяет предсказывать по 2 матча в день

стандартная – действует 30 дней и позволяет предсказывать до 20 матчей в день

золотая – действует 30 дней и позволяет предсказывать до 50 матчей в день

платиновая – действует 30 дней и позволяет предсказывать до 100 матчей в день

К курсовому проекту предъявляются следующие требования:

1)Должна быть создана система авторизации пользователей.

2)Должна быть создана система разграничения пользователей в правах.

3)У пользователя должна быть возможность купить подписку.

4)Пользователь должен иметь возможность просмотреть все типы подписок и выбрать подходящую.

5)Пользователей должен иметь возможность спрогнозировать результат выбранного матча

6)Пользователь должен иметь возможность просмотреть свои прогнозы

7)Пользователь должен видеть информацию о количестве оставшихся на день прогнозов.

8)Пользователь должен иметь возможность отличать спрогнозированные им матчи от не спрогнозированных.

9)Суперпользователь должен иметь возможность просмотреть график с количеством активных, закрытых и общих подписок пользователей.

**Проектирование проекта**

Ниже приведена общая схема архитектуры проекта

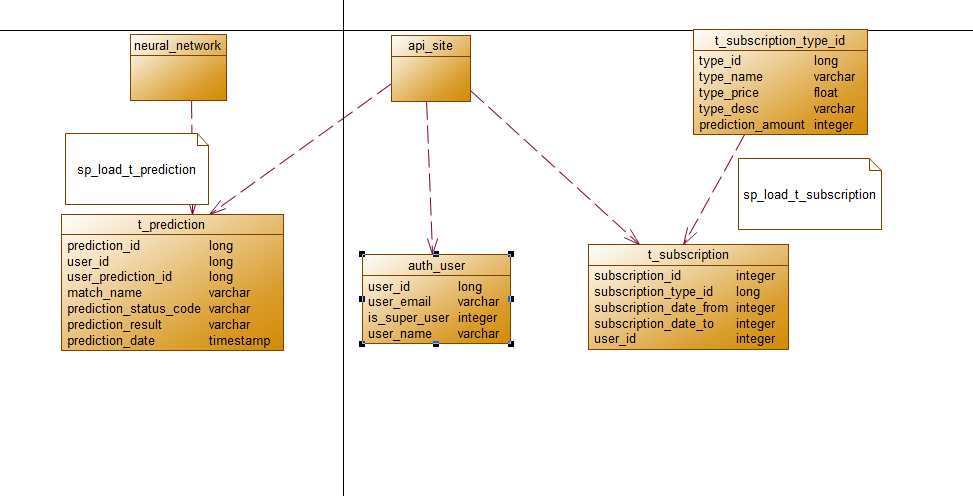


Рис 1. общая схема архитектуры проекта

Дадим пояснение архитектуре проекта: основные сущности — это тип подписки, подписка, пользователь, прогноз пользователя.

Поля типа подписки:

1)type\_id уникальный номер типа подписки.

2)type\_name: имя типа подписки: trial, standard, gold, platinum.

3)type\_price: цена типа подписки: 0, 4.99, 9.99, 17.99.

4)type\_desc: расширенное описание подписки: “can be your first subscription, free for 7 days, 2 prediction per day”, “valid for 30 days, 20 predictions per day”,

“valid for 30 days, 50 predictions per day”, “valid for 30 days, 100 predictions per day”.

5)prediction amount: число прогнозов, которые может делать пользователь с этим типом подписки: 2, 20, 50, 100

Поля подписки:

1)subscription\_id: уникальный номер подписки.

2)subscription\_type\_id: внешний ключ на таблицу subscription\_type\_id

3)subscription\_date\_from: дата старта подписки

4) subscription\_date\_to: дата окончания подписки

5) user\_id: внешний ключ на таблицу auth\_user

Поля пользователя:

1)user\_id: уникальный номер пользователя.

2)user\_email: электронная почта пользователя.

3)user\_name: имя пользователя.

4)is\_super\_user: принимает значения 1 и 0, влияет на то, может ли пользователь видеть часть сервиса по анализу пользователей и их подписок

Поля прогнозирования:

1)prediction\_id: уникальный номер прогнозирования.

2)user\_id: внешний ключ на таблицу auth\_user.

3)user\_prediction\_id: определяет номер прогнозирования конкретного пользователя

4)match\_name: имя матча, результат которого прогнозируется.

5)prediction\_status\_code: код статуса результата прогнозирования: “success” – успешный прогноз, этот код прогноза отдает нейронная сеть, “in progress” – прогноз в процессе, этот код прогноза приходит из сервиса, “failure” – ошибка в прогнозе, этот код прогноза отдает нейронная сеть.

6)prediction result: строка в виде один из спортсменов победил с такой-то вероятностью.

7) prediction\_date: дата, когда пользователь совершил прогноз.

Для работы с сущностями будет реализован механизм авторизации пользователя.

Будут реализованы процедуры по вставке новых подписок, прогнозов.

Будут реализованы процедуры по выводу типов подписок, по выводу прогнозов пользователя, по выводу оставшегося дневного количества прогнозов пользователя.

Будет реализована процедура по анализу количества открытых, закрытых и общих подписок пользователей.

**Программная реализация**

Для построения архитектуры была выбрана база данных Oracle.

Oracle Database — это объектно-реляционная система управления базами данных (СУБД) от компании Oracle.

На мировом рынке корпоративных систем управления базами данных (СУБД) доминирующее положение занимает традиционная тройка продуктов: IBM DB2, Microsoft SQL Server и Oracle. Более 80% рынка СУБД в течение долгих лет контролируется тремя компаниями производителями: IBM, Oracle и Microsoft. По статистическим данным на рынке России лидирующее положение занимает Oracle, так как по статистическим данным за 2017 год, данная СУБД занимает более 60% всего рынка, среди других СУБД и около 30% мирового рынка СУБД (рис.2).

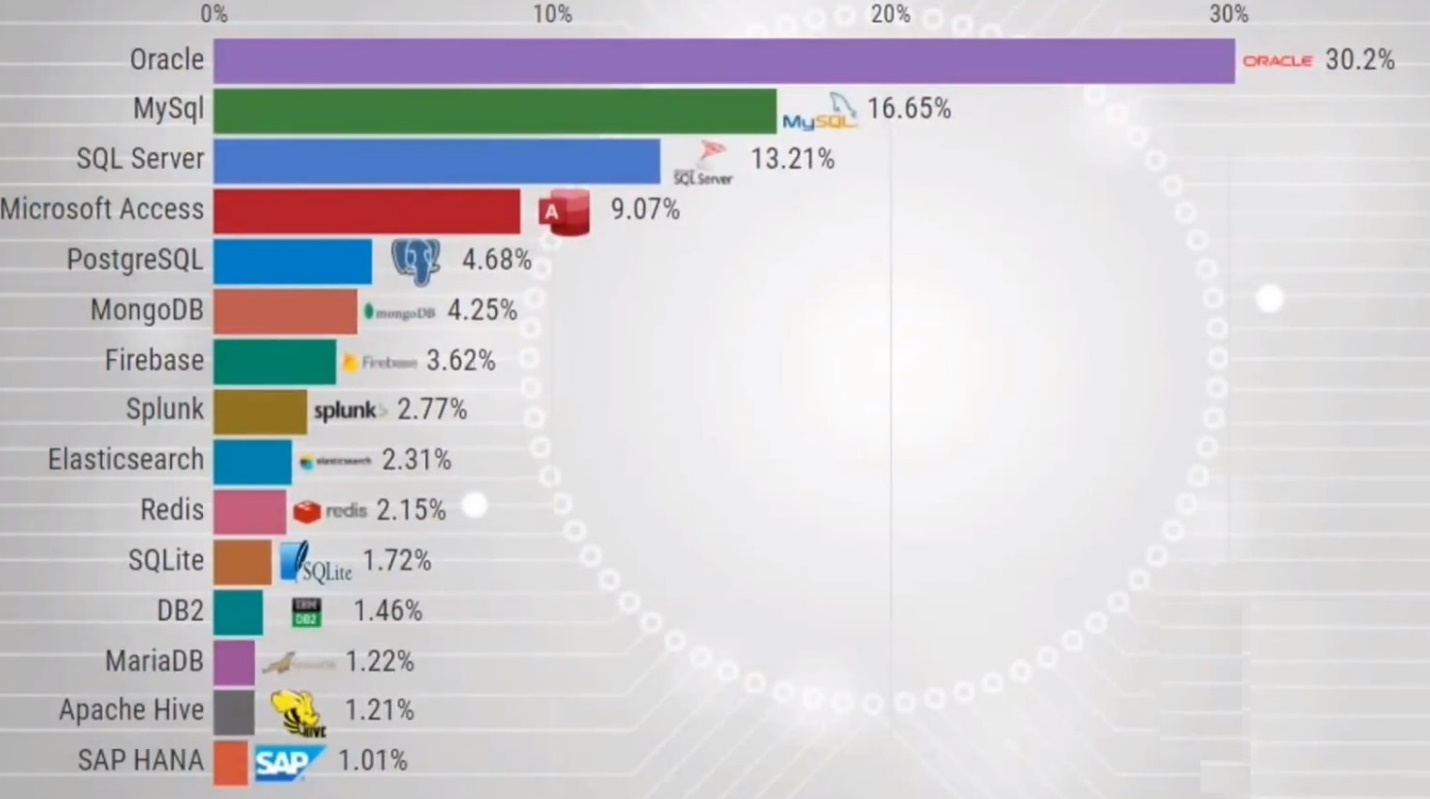


Рис 2. Инфографик популярности СУБД на мировом рынке

На сегодняшний день СУБД Oracle поддерживают свыше 80 вариантов операционной среды в широком диапазоне, включая мэйнфреймы IBM, мини-компьютеры DEC VAX, UNIX, Windows и множество других платформ.

Данная СУБД имеет массу преимуществ.  
  
Oracle поддерживает самые большие базы данных. Большое количество пользователей для этой системы также не помеха. СУБД способна поддерживать любых пользователей, в любом количестве, которые при этом одновременно выполняют разные задачи. В Oracle не происходит соперничества между разными видами данных.  
  
СУБД Oracle хорошо обрабатывает транзакции. Система сохраняет высокую производительность, в результате чего пользователи не страдают от низкой скорости обработки.  
  
Система обладает высокой степенью готовности. В разных установках продолжительность работы Oracle индивидуальная. Так, например, в некоторых, система способна работать круглосуточно. При этом откат БД или какие-либо сбои системы не приводят к остановке работы базы.

Рассмотрим теперь построение архитектуры БД под нашу задачу.

Построение архитектуры будет включать в себя 3 этапа:

1) Создание таблиц

2) Первичная инициализация данных в таблицах

3) Создание процедур, удовлетворяющих требованиям проекта

Рассмотрим первый этап.

На первом этапе были созданы следующие таблицы:

1) t\_subscription\_type

2) t\_subscription

3) t\_prediction

4) auth\_user

Таблица t\_subscription\_type содержит следующие колонки:

1) type\_id с типом number

2) type\_name с типом varchar2(500)

3) type\_desc с типом varchar2(2000)

4) type\_price с типом decimal(5,2)

5) prediction\_amount с типом number

Также на колонку type\_id было наложено ограничение первичного ключа.

Таблица t\_subscription содержит следующие колонки:

1) subscription\_id с типом number

2) subscription\_type\_id с типом number

3) user\_id с типом number

4) subscription\_start\_date с типом number

5) subscription\_end\_date с типом number

Также на колонку subscription\_id было наложено ограничение первичного ключа.

Таблица t\_prediction содержит следующие колонки:

1) prediction\_id с типом number

2) user\_prediction\_id с типом number

3) user\_id с типом number

4) match\_name с типом varchar2(500)

5) prediction\_status\_code с типом varchar2(500)

6) prediction\_result с типом varchar2(500)

7) prediction\_date с типом number

Также на колонку prediction\_id было наложено ограничение первичного ключа.

Таблица auth\_user была создана ORM оболочкой фрэймворка Django, на котором был написан сервис по прогнозу результат спортивных матчей.

Таблица auth\_user содержит следующие колонки:

1) user\_id с типом number

2) user\_name с типом varchar2(500)

3) user\_email с типом varchar2(500)

4) is\_super\_user с типом number

Рассмотрим теперь инициализацию таблиц первичными данными:

В t\_subscription\_type в качестве необходимых данных были вставлены следующие:

1) type\_id = 1, type\_name = ‘Trial’, type\_desc = ‘can be your first subscription, free for 7 days, 2 prediction per day’, type\_price = 0, prediction\_amount = 2

2) type\_id = 2, type\_name = ‘Standart, type\_desc = ‘valid for 30 days, 20 prediction per day’, type\_price = 4.99, prediction\_amount = 20

3) type\_id = 3, type\_name = ‘Gold, type\_desc = ‘valid for 30 days, 50 prediction per day’, type\_price = 9.99, prediction\_amount = 50

4) type\_id = 4, type\_name = ‘Platinum, type\_desc = ‘valid for 30 days, 100 prediction per day’, type\_price = 17.99, prediction\_amount = 100

В auth\_user в качестве служебных данных были вставлены следующие:

1) user\_id = 1, user\_name = ‘userr’, user\_email=’userr@gmail.com’, is\_super\_user =1

2) user\_id = 21, user\_name = ‘anton\_dwh, user\_email=’anton\_dwh@mail.ru’, is\_super\_user =0

3) user\_id = 41, user\_name = ‘dwh, user\_email=’dwh@gmail.com’, is\_super\_user =0

В t\_subscription в качестве тестовых данных были вставлены следующие:

1) subscription\_id = 1, subscription\_type\_id = 3, user\_id=1, subscription\_start\_date=20221023, subscription\_end\_date=20221030

2) subscription\_id = 2, subscription\_type\_id = 1, user\_id=21, subscription\_start\_date=20221023, subscription\_end\_date=20221122

В t\_prediction в качестве тестовых данных были вставлены следующие:

1) prediction\_id = 1, user\_id = 1, user\_prediction\_id = 1, match\_name=’Arneodo R / Weissborn T-S’, prediction\_status\_code = ‘success’, prediction\_result=’Arneodo R win with probability 70%’, prediction\_date = 20221023

2) prediction\_id = 2, user\_id = 1, user\_prediction\_id = 2, match\_name=’ Ritschard A / Martineau S’, prediction\_status\_code = ‘success’, prediction\_result=’Ritschard A win with probability 90%’, prediction\_date = 20221023

3) prediction\_id = 3, user\_id = 1, user\_prediction\_id = 3, match\_name=’Arneodo R / Weissborn T-S’, prediction\_status\_code = ‘in progress’, prediction\_result=‘in progress’, prediction\_date = 20221023

4) prediction\_id = 4, user\_id = 21, user\_prediction\_id = 1, match\_name=’Arneodo R / Weissborn T-S’, prediction\_status\_code = ‘in progress’, prediction\_result=‘in progress’, prediction\_date = 20221023

5) prediction\_id = 5, user\_id = 1, user\_prediction\_id = 4, match\_name=’Arneodo R / Weissborn T-S’, prediction\_status\_code = ‘in progress’, prediction\_result=‘in progress’, prediction\_date = 20221024

6) prediction\_id = 6, user\_id = 21, user\_prediction\_id = 5, match\_name=’Arneodo R / Weissborn T-S’, prediction\_status\_code = ‘in progress’, prediction\_result=‘in progress’, prediction\_date = 20221028

7) prediction\_id = 7, user\_id = 21, user\_prediction\_id = 1, match\_name=’Arneodo R / Weissborn T-S’, prediction\_status\_code = ‘in progress’, prediction\_result=‘in progress’, prediction\_date = 20221028

8) prediction\_id = 8, user\_id = 1, user\_prediction\_id = 1, match\_name=’Arneodo R / Weissborn T-S’, prediction\_status\_code = ‘in progress’, prediction\_result=‘in progress’, prediction\_date = 20221029

Рассмотрим этап создания процедур, ниже будут приведены список имен созданных процедур, параметры, которые они принимают, результаты, которые они возвращают и какой цели эти процедуры служат.

1) SP\_GET\_SUBSCRIPTION\_TYPES - принимает на вход два параметра p\_user\_id – id пользователя и p\_results – курсор, в который она отдаст результат. Возвращает типы подписок, которые есть на сервисе. Нужна для того, чтобы пользователь мог увидеть, какие подписки есть на сервисе.

2) SP\_GET\_PREDICTIONS - принимает на вход два параметра p\_user\_id – id пользователя и p\_results – курсор, в который она отдаст результат. Возвращает прогнозы определенного пользователя. Нужна для того, чтобы пользователь сервиса мог увидеть свои прогнозы.

3) SP\_GET\_USER\_EMAIL - принимает на вход два параметра p\_user\_id – id пользователя и p\_results – курсор, в который она отдаст результат. Возвращает id и email определенного пользователя. Нужна для того, чтобы пользователь сервиса мог увидеть свой email, под которым он был зарегистрирован в системе.

4) SP\_GET\_SUBSCRIPTION\_TYPES\_BY\_ID - принимает на вход два параметра p\_type\_id – id типа подписки и p\_results – курсор, в который она отдаст результат. Возвращает информации об определенном типе подписок, которые есть на сервисе. Нужна для того, чтобы пользователь мог перейти на страницу подтверждения покупки выбранной подписки.

5) SP\_LOAD\_NEW\_SUBSCRIPTION – принимает на вход три параметра: p\_user\_id – id пользователя, p\_subscription\_type\_id – id типа подписки, p\_days – количество дней, сколько будет длиться подписка, он может принимать значения 7 или 30 в зависимости от типа подписки. Вставляет в t\_subscription следующие поля:

1)subscription\_id = max(subscription\_id) in t\_subscription + 1

2)subscription\_type\_id = p\_subscription\_type\_id

3)user\_id = p\_user\_id

4)subscription\_start\_date = текущее системное время приведенное к числу

5)subscription\_end\_date = текущее системное время + p\_days приведенное к числу

Привести текущее системное время к числу можно следующей функцией:

TO\_NUMBER(TO\_CHAR(CURRENT\_TIMESTAMP, ‘YYYYMMDD’))

Данная процедура используется для вставки в таблицу подписок новой подписки, которую приобрел пользователь.

6) SP\_LOAD\_NEW\_USER\_PREDICTION – принимает на вход два параметра: p\_user\_id – id пользователя, p\_match\_name – имя матча. Вставляет в таблицу t\_prediction следующие поля:

1)prediction\_id = max(prediction\_id) in t\_prediction + 1

2)user\_id = p\_user\_id

3) user\_prediction\_id = max(prediction\_id) in t\_prediction where user\_id = p\_user\_id + 1

4)match\_name = p\_match\_name

5)prediction\_status\_code = ‘in progress’

6)prediction\_result = ‘in progress’

7)prediction\_date = текущее системное время приведенное к числу

Данная процедура используется для вставки в таблицу прогнозов нового прогноза, который решил сделать пользователь.

7) SP\_GET\_SUBSCRIPTION\_ANALYSIS – принимает на вход три параметра, p\_start\_date – время начала промежутка дат для анализа, p\_end\_date – время конца промежутка дат для анализа, p\_results – курсор, в который данная процедура отдаст свой результат. Она возвращает число общих подписок, стартовавших подписок и закончившихся подписок в рамках одного дня.

Она работает следующим образом: сначала мы фильтруем число подписок по двум условиям:

1) subscription\_start\_date <= p\_end\_date

2) subscription\_end\_date >= p\_start\_date

Затем мы группируем подписки по дню и считаем:

1) count(subscription\_id) в качестве общих подписок

2) count(subscription\_id) where subscription\_start\_date = some\_date в качестве числа стартовавших подписок

3) count(subscription\_id) where subscription\_end\_date = some\_date в качестве числа закрывшихся подписок

Данная процедура нужна для вывода данных на график анализа в сервисе по прогнозу спортивных данных.

8) SP\_GET\_CURRENT\_USER\_PREDICTIONS\_AMOUNT – принимает на вход три параметра, p\_user\_id – уникальный id пользователя, p\_results – курсор, в который данная процедура отдаст свой результат. Она возвращает число прогнозов пользователя, оставшихся у него на текущий день с учетом типа его подписки.

Она работает следующим образом:

1)берем последнюю подписку пользователя, которую он приобрел, то есть его текущую подписку

2)проверяем, что она не истекла

3)берем максимальное число прогнозов в день согласно типу подписки

4)находим через count(prediction\_id) число прогнозов пользователя, которые он совершил в текущий день

5) вычитаем 4 из 3 и получаем количество подписок пользователя, которые у него остались

Данная процедура нужна для вывода пользователю информации о количестве оставшихся у него на текущий день прогнозов и для решения позволять ли еще делать пользователю прогнозы или нет.

9) SP\_GET\_CURRENT\_USER\_SUBSCRIPTION – данная процедура принимает на вход два параметра, p\_user\_id – уникальный id пользователя, p\_results – курсор, в который данная процедура отдаст свой результат.

Она возвращает тип текущей подписки пользователя.

Данная процедура работает следующим образом:

1) выбираем последнюю подписку, которую приобрел пользователь

2) выбираем её тип

Данная процедура используется для показа на сервисе о том, какая именно подписка пользователя является у нее текущей.

**Тестирование проекта**

Проверим работу написанных процедур.

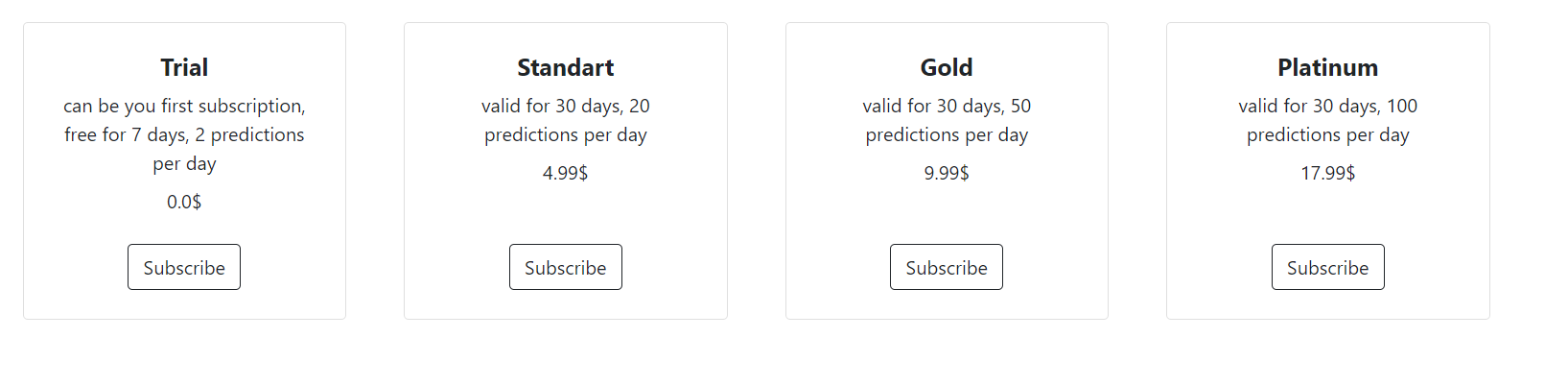
Тест № 1:

Выполним на сервисе вызов процедуры SP\_GET\_SUBSCRIPTION\_TYPES

cursor.callproc("SP\_GET\_SUBSCRIPTION\_TYPES", (2, ref\_cursor))

Ожидаем увидеть все 4 типа подписок.

Проверяем результат и видим следующее.

 Рис 3. Результат вызова процедуры SP\_GET\_SUBSCRIPTION\_TYPES

Видим, что нам вернулись верные данные.

Тест № 1 пройден.

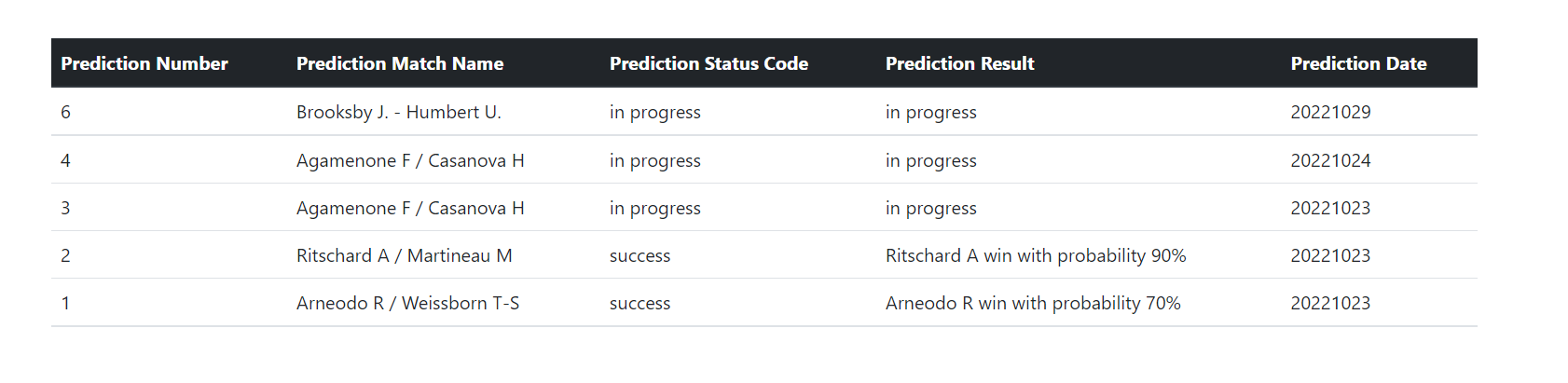
Тест № 2:

Выполним на сервисе вызов процедуры SP\_GET\_PREDICTIONS

cursor.callproc("SP\_GET\_PREDICTIONS", (1, ref\_cursor))

Ожидаем увидеть все прогнозы первого пользователя.

Проверяем результат и видим следующее.

 Рис 4. Результат вызова процедуры SP\_GET\_PREDICTIONS

Видим, что нам вернулись верные данные.

Тест № 2 пройден.

Тест № 3:

Выполним на сервисе вызов процедуры SP\_GET\_USER\_EMAIL

cursor.callproc("SP\_GET\_USER\_EMAIL", (1, ref\_cursor))

Ожидаем увидеть email первого пользователя.

Проверяем результат и видим следующее.

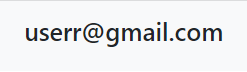


Рис 5. Результат вызова процедуры SP\_GET\_USER\_EMAIL

Видим, что нам вернулись верные данные.

Тест № 3 пройден.

Тест № 4:

Выполним на сервисе вызов процедуры SP\_GET\_SUBSCRIPTION\_TYPES\_BY\_ID

cursor.callproc("SP\_GET\_SUBSCRIPTION\_TYPES\_BY\_ID", (2, ref\_cursor))

Ожидаем увидеть информацию по подписке типа 2.

В сервисе это реализовано следующим образом, в url добавляется id подписки.

Проверяем результат и видим следующее.



Рис 6. Результат вызова SP\_GET\_SUBSCRIPTION\_TYPES\_BY\_ID

Видим, что нам вернулись верные данные.

Тест № 4 пройден.

Тест № 5:

Выполним на сервисе вызов процедуры SP\_LOAD\_NEW\_SUBSCRIPTION

cursor.callproc("SP\_LOAD\_NEW\_SUBSCRIPTION", (1, 2, 30))

Ожидаем что у пользователя поменяется тип текущей подписки.

До вызова процедуры видим следующее.

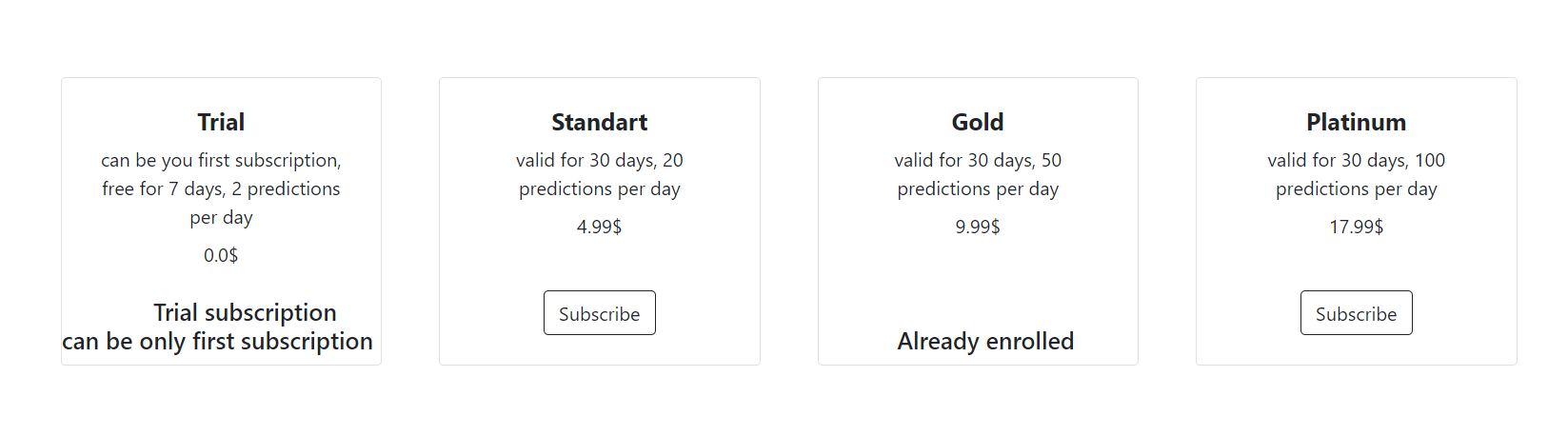


Рис 7. Подписка пользователя до вызова SP\_LOAD\_NEW\_SUBSCRIPTION

После вызова видим следующее.

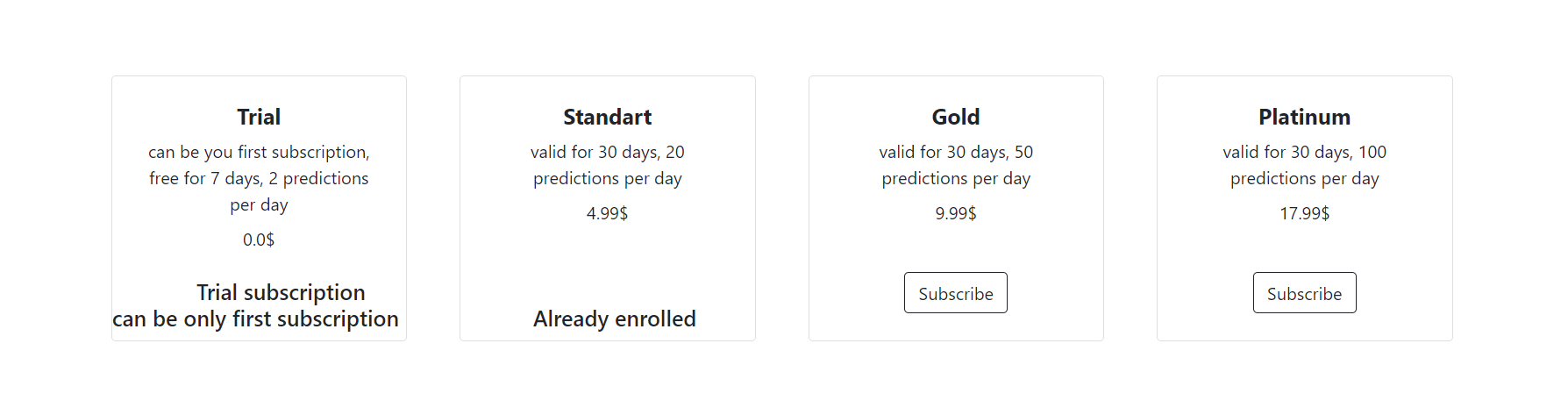


Рис 8. Результат вызова SP\_LOAD\_NEW\_SUBSCRIPTION

Видим, что подписка обновилась корректно.

Тест № 5 пройден.

Тест № 6:

Выполним на сервисе вызов SP\_LOAD\_NEW\_USER\_PREDICTION

cursor.callproc("SP\_LOAD\_NEW\_USER\_PREDICTION ", (1, ‘de Minaur A. – Rune H.’))

Ожидаем что у пользователя обновятся текущие прогнозы.

До вызова процедуры видим следующее.

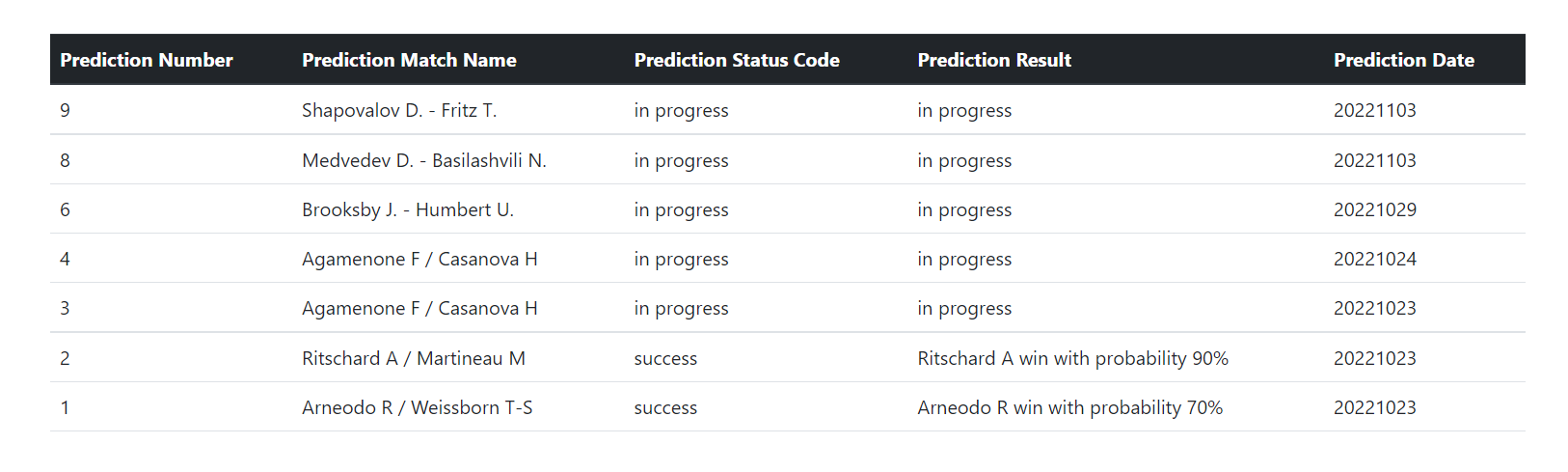


Рис 9. Прогнозы пользователя до вызова SP\_LOAD\_NEW\_USER\_PREDICTION

После вызова видим следующее.

Изображение выглядит как текст, снимок экрана, внутренний, ноутбук

Автоматически созданное описание

Рис 10. Прогнозы пользователя после вызова SP\_LOAD\_NEW\_USER\_PREDICTION

Видим, что прогнозы обновились корректно.

Тест № 6 пройден.

Тест № 7:

Выполним на сервисе вызов SP\_GET\_SUBSCRIPTION\_ANALYSIS

cursor.callproc("SP\_GET\_SUBSCRIPTION\_ANALYSIS", (20221023, 20221031, ref\_cursor))

Ожидаем что отобразится график с корректной информацией о подписках.

В результате видим следующее:

Изображение выглядит как стол

Автоматически созданное описание

Рис 11. График SP\_GET\_SUBSCRIPTION\_ANALYSIS

Видим, что вывелись корректные данные.

Тест № 7 пройден.

Тест № 8:

Выполним на сервисе вызов SP\_GET\_CURRENT\_USER\_PREDICTIONS\_AMOUNT

cursor.callproc("SP\_GET\_CURRENT\_USER\_PREDICTIONS\_AMOUNT", (1, ref\_cursor))

Ожидаем что будет выведена корректная информация об оставшихся на текущий день у пользователя прогнозах.

В результате видим следующее:

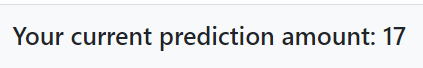


Рис 12. Результат вызова процедуры SP\_GET\_CURRENT\_USER\_PREDICTIONS\_AMOUNT

Видим, что вывелись корректные данные.

Тест № 8 пройден.

Тест № 9:

Выполним на сервисе вызов SP\_GET\_CURRENT\_USER\_SUBSCRIPTION

cursor.callproc("SP\_GET\_CURRENT\_USER\_SUBSCRIPTION", (1, ref\_cursor))

Ожидаем, что будет выведена корректная информация о текущей подписке пользователя.

В результате видим следующее:

Изображение выглядит как текст

Автоматически созданное описание

Рис 13. Результат вызова SP\_GET\_CURRENT\_USER\_SUBSCRIPTION

Видим, что вывелись корректные данные.

Тест № 9 пройден.

Все тесты пройдены и всё работает верно. Тестирование проекта окончено.

**Заключение**

В результате работы над курсовом проектом была создана архитектура базы данных для работы с пользователями, их подписками и прогнозами на сервисе по прогнозированию результатов спортивных матчей.

Был произведен анализ бизнес-требований сервиса, бизнес-требования сервиса были преобразованы в конкретный технический замысел.

Были созданы таблицы, подходящие техническим требованиям сервиса.

Была произведена инициализация таблиц первичными данными.

Наконец был написан ряд процедур, которые на основе данных, загруженных в таблицы, отдавали необходимый их задумке результат в сервис.

Данная архитектура была вручную протестирована в разделе тестирование проекта.

Результаты всех тестов оказались удовлетворительными.

На основе результатов тестов можно утверждать, что построение архитектуры прошло успешно и полностью удовлетворяет бизнес-требованиям.

В качестве перспектив дальнейшего развития курсового проекта можно отметить следующие.

Во-первых, можно подумать над интеграцией текущей архитектуры с платежной системой stripe. Данная платежная система является одной из самых популярных в мире. Интеграция данной системы в текущую систему позволило бы сделать проект более живым и подходящим под реальные практические требования.

Во-вторых, стоит подумать о расширении функционала путем написания новых процедур на основе текущей архитектуры.

**Список использованных источников**

1. Создание базы данных «Больница» [Электронный ресурс]. – 2013. – Режим доступа: <https://www.skachatreferat.ru/referaty/%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-%D0%91%D0%B0%D0%B7%D1%8B-%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D1%85-%D0%91%D0%BE%D0%BB%D1%8C%D0%BD%D0%B8%D1%86%D0%B0/294423.html> Дата доступа: 29.10.2022

2. Область применения баз данных [Электронный ресурс]. – 2016. – Режим доступа: <https://scienceforum.ru/2016/article/2016020613/> Дата доступа: 29.10.2022

3. Простая транзакция базы данных Oracle [Электронный ресурс]. – 2014. – Режим доступа: <https://oracle-dba.ru/docs/architecture/transactions/simple-transaction/> Дата доступа: 30.10.2022

4. Архитектура СУБД Oracle [Электронный ресурс]. – 2017. – Режим доступа: <https://oracle-patches.com/oracle/begin/%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0-%D1%81%D1%83%D0%B1%D0%B4-oracle> Дата доступа: 31.10.2022

5. Область применения баз данных [Электронный ресурс]. – 2014. – Режим доступа: <http://ora-sql.ru/arxitektura-pamyati-oracle.html> Дата доступа: 28.10.2022

**Приложение А**

**(обязательное)**

**Исходный код**

drop table t\_subscription\_type;

create table t\_subscription\_type (

type\_id number,

type\_name varchar2(500),

type\_desc varchar2(2000),

type\_price decimal (5,2),

prediction\_amount number,

constraint pk\_type\_id primary key (type\_id)

);

insert into t\_subscription\_type

(

type\_id,

type\_name,

type\_desc,

type\_price,

prediction\_amount

)

values ( 1, 'Trial', 'can be you first subscription, free for 7 days, 2 predictions per day', 0, 2);

insert into t\_subscription\_type

(

type\_id,

type\_name,

type\_desc,

type\_price,

prediction\_amount

)

values (2, 'Standart', 'valid for 30 days, 20 predictions per day', 4.99, 20);

insert into t\_subscription\_type

(

type\_id,

type\_name,

type\_desc,

type\_price,

prediction\_amount

)

values (3, 'Gold','valid for 30 days, 50 predictions per day', 9.99, 50);

insert into t\_subscription\_type

(

type\_id,

type\_name,

type\_desc,

type\_price,

prediction\_amount

)

values (4, 'Platinum','valid for 30 days, 100 predictions per day', 17.99, 100);

drop table t\_prediction

create table t\_prediction (

prediction\_id number,

user\_id number,

user\_prediction\_id number,

match\_name varchar2(500),

prediction\_status\_code varchar2(500),

prediction\_result varchar2(500),

prediction\_date number,

constraint pk\_prediction\_id primary key (prediction\_id)

);

create table t\_subscription(

subscription\_id number,

subscription\_type\_id number,

user\_id number,

subscription\_start\_date number,

subscription\_end\_date number,

constraint pk\_subscription\_id primary key (subscription\_id)

);

CREATE OR REPLACE PROCEDURE SP\_GET\_SUBSCRIPTION\_TYPES(type\_id number, p\_results OUT SYS\_REFCURSOR)

AS

BEGIN

OPEN p\_results FOR

SELECT type\_id,

type\_name,

type\_desc,

type\_price,

prediction\_amount

FROM t\_subscription\_type

ORDER BY type\_id;

END;

/

CREATE OR REPLACE PROCEDURE SP\_GET\_PREDICTIONS(p\_user\_id number, p\_results OUT SYS\_REFCURSOR)

AS

BEGIN

OPEN p\_results FOR

SELECT

prediction\_id,

user\_id,

user\_prediction\_id,

match\_name,

prediction\_status\_code,

prediction\_result,

prediction\_date

FROM t\_prediction

WHERE user\_id = p\_user\_id

ORDER BY user\_prediction\_id DESC;

END;

/

CREATE OR REPLACE PROCEDURE SP\_GET\_USER\_EMAIL(p\_user\_id number, p\_results OUT SYS\_REFCURSOR)

AS

BEGIN

OPEN p\_results FOR

SELECT

id,

email

FROM auth\_user

WHERE id = p\_user\_id;

END;

/

CREATE OR REPLACE PROCEDURE SP\_GET\_SUBSCRIPTION\_TYPES\_BY\_ID(p\_type\_id number, p\_results OUT SYS\_REFCURSOR)

AS

BEGIN

OPEN p\_results FOR

SELECT type\_id,

type\_name,

type\_desc,

type\_price,

prediction\_amount

FROM t\_subscription\_type

WHERE type\_id = p\_type\_id;

END;

/

CREATE OR REPLACE PROCEDURE SP\_LOAD\_NEW\_SUBSCRIPTION(p\_user\_id number, p\_subscription\_type\_id number, p\_days number)

AS

BEGIN

INSERT INTO t\_subscription

(

subscription\_id,

subscription\_type\_id,

user\_id,

subscription\_start\_date,

subscription\_end\_date

)

VALUES (

(select nvl(max(subscription\_id),0) from t\_subscription) + 1,

p\_subscription\_type\_id,

p\_user\_id,

TO\_CHAR(CURRENT\_TIMESTAMP, 'YYYYMMDD'),

TO\_CHAR(CURRENT\_TIMESTAMP + p\_days, 'YYYYMMDD')

);

END;

/

CREATE OR REPLACE PROCEDURE SP\_LOAD\_NEW\_USER\_PREDICTION(p\_match\_name varchar2, p\_user\_id number)

AS

BEGIN

INSERT INTO t\_prediction

(

prediction\_id,

user\_id,

user\_prediction\_id,

match\_name,

prediction\_status\_code,

prediction\_result,

prediction\_date

)

VALUES (

(select nvl(max(prediction\_id),0) from t\_prediction) + 1,

p\_user\_id,

(select nvl(max(user\_prediction\_id),0) from t\_prediction where user\_id = p\_user\_id) + 1,

p\_match\_name,

'in progress',

'in progress',

TO\_CHAR(CURRENT\_TIMESTAMP, 'YYYYMMDD')

);

END;

/

CREATE OR REPLACE PROCEDURE SP\_GET\_SUBSCRIPTION\_ANALYSIS(p\_subscription\_id varchar2, p\_results OUT SYS\_REFCURSOR)

AS

BEGIN

OPEN p\_results FOR

SELECT

c.number\_date,

COUNT(t.subscription\_id) as total\_subs,

COUNT(CASE WHEN t.subscription\_start\_date = c.number\_date then t.subscription\_id else null end) as start\_subs,

COUNT(CASE WHEN t.subscription\_end\_date = c.number\_date then t.subscription\_id else null end) as end\_subs

FROM t\_subscription t

JOIN calendar c on c.number\_date between t.subscription\_start\_date and t.subscription\_end\_date

GROUP BY c.number\_date;

END;

/

CREATE OR REPLACE PROCEDURE SP\_GET\_CURRENT\_USER\_PREDICTIONS\_AMOUNT(p\_user\_id number, p\_results OUT SYS\_REFCURSOR)

AS

BEGIN

OPEN p\_results FOR

with cte\_user\_subscription as (

select user\_id,

max(subscription\_id) as subscription\_id

from t\_subscription

where user\_id = p\_user\_id

group by user\_id

)

, cte\_subscription\_user\_info as (

select t.user\_id,

t.subscription\_id,

t.subscription\_type\_id,

t.subscription\_start\_date,

t.subscription\_end\_date

from t\_subscription t

join cte\_user\_subscription c on c.subscription\_id = t.subscription\_id

where to\_number(to\_char(current\_timestamp, 'YYYYMMDD')) between t.subscription\_start\_date and t.subscription\_end\_date

)

SELECT nvl(MAX(tst.PREDICTION\_AMOUNT), 0) - COUNT(t.PREDICTION\_ID) as PREDICTION\_COUNT

FROM cte\_subscription\_user\_info c

LEFT JOIN t\_prediction t on c.USER\_ID = t.USER\_ID and t.PREDICTION\_DATE = to\_number(to\_char(current\_timestamp, 'YYYYMMDD'))

LEFT JOIN t\_subscription\_type tst on c.subscription\_type\_id = tst.type\_id;

END;

/

CREATE OR REPLACE PROCEDURE SP\_GET\_CURRENT\_USER\_SUBSCRIPTION(p\_user\_id number, p\_results OUT SYS\_REFCURSOR)

AS

BEGIN

OPEN p\_results FOR

WITH cte\_max\_subscription as (

select MAX(SUBSCRIPTION\_ID) as SUBSCRIPTION\_ID

from t\_subscription

where user\_id = p\_user\_id

)

SELECT nvl(t.SUBSCRIPTION\_TYPE\_ID, -1) as SUBSCRIPTION\_TYPE\_ID

FROM cte\_max\_subscription c

LEFT JOIN t\_subscription t on c.SUBSCRIPTION\_ID = t.SUBSCRIPTION\_ID;

END;

/