МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

Специальность 1-40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка базы данных аукционной площадки»

Выполнил студент Cавин Роман Алексеевич

(Ф.И.О.)

Руководитель проекта Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: Блинова Е.А.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

**Содержание**

[Реферат 3](#_Toc59113165)

[1. Аналитический обзор литературы 4](#_Toc59113166)

[2. Архитектура базы данных 5](#_Toc59113167)

[2.1. Таблицы базы данных 5](#_Toc59113168)

[2.2. Процедуры для решения поставленных задач 6](#_Toc59113169)

[2.3. Индекс базы данных 8](#_Toc59113170)

[3. Реализация задач 8](#_Toc59113171)

[3.1. Регистрация и авторизация пользователей 8](#_Toc59113173)

[3.2. Добавление новых учетных записей пользователя 9](#_Toc59113174)

[3.3. Добавление лотов 10](#_Toc59113175)

[4. Импорт и экспорт данных. 12](#_Toc59113176)

[5. Описание технологии 14](#_Toc59113177)

[6. Тестирование 18](#_Toc59113178)

[Заключение 20](#_Toc59113179)

[Список использованных источников 21](#_Toc59113180)

[ПРИЛОЖЕНИЕ A 22](#_Toc59113181)

[ПРИЛОЖЕНИЕ B 24](#_Toc59113182)

[ПРИЛОЖЕНИЕ C 25](#_Toc59113183)

# Реферат

Курсовая работа изложена на 36 страницах, которые включают в себя 6 разделов, 6 подразделов, введение, архитектуру, реализацию, тестирование, заключение, используемую литературу и приложения.

Цель курсового проекта — разработка базы данных приложения, которая обеспечит простое и удобное взаимодействие между будущим приложением и собой, что позволит пользователю купить предмет, а также узнать информацию об интересующих предметах и аукционах, их стоимости и полной информации о соответствующем предмете.

Используемая СУБД – Oracle Database 12c.

Изучаемая технология – Резервное копирование и восстановление.

В первой главе проводится аналитический обзор схожих приложений по тематике курсового проекта.

Вторая глава посвящена процессу проектирования системы и описанию технологий, использованных во время выполнения проекта.

В третьей главе описан процесс выполнения поставленных задач в системе.

В четвертой главе описан процесс импорта и экспорта данных.

В пятой главе описано выполнение описанной в задании технологии в базе данных.

В шестой главе описано тестирование и анализ данных системы.

В заключении приведены результаты проделанной работы

## Аналитический обзор литературы

Copart – аукцион авто — программа, которая реализует онлайн-аукцион автомобилей. (рис 1.1). Данное приложение позволяет пользователям участвовать в проводимых аукционах, на которых представлены подержанные автомобили. Пользователи могут получать автомобили фильтруя их по году, по типу, марке, модели и местоположению, а так же получать информацию о транспортных средствах, внесенных в список отслеживаемых позиций.

Из преимуществ приложения можно выделить широкий спектр фильтров, возможность отслеживать местоположение авто и добавлять авто в список отслеживаемых. Из недостатков – узконаправленность аукциона.

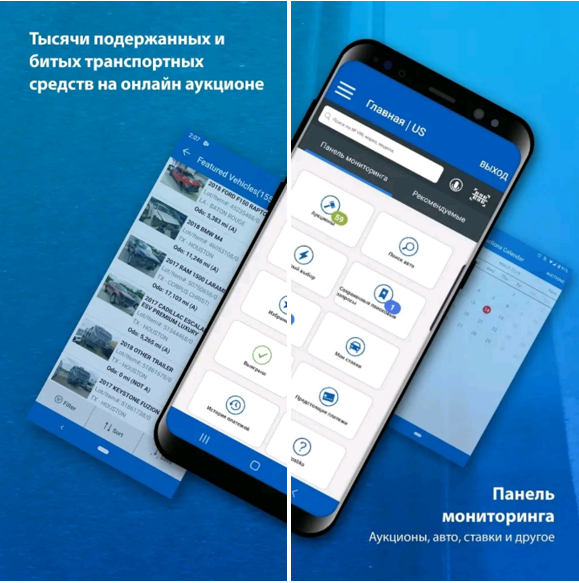


Рисунок 1.1 – Возможности Copart

## Архитектура базы данных

## Таблицы базы данных

Для хранения и предоставления данных пользователю, была разработана база данных, которая содержит в себе 7 основных таблиц, все связаны друг с другом внешними ключами. Диаграмма представлена на рисунке 2.1.

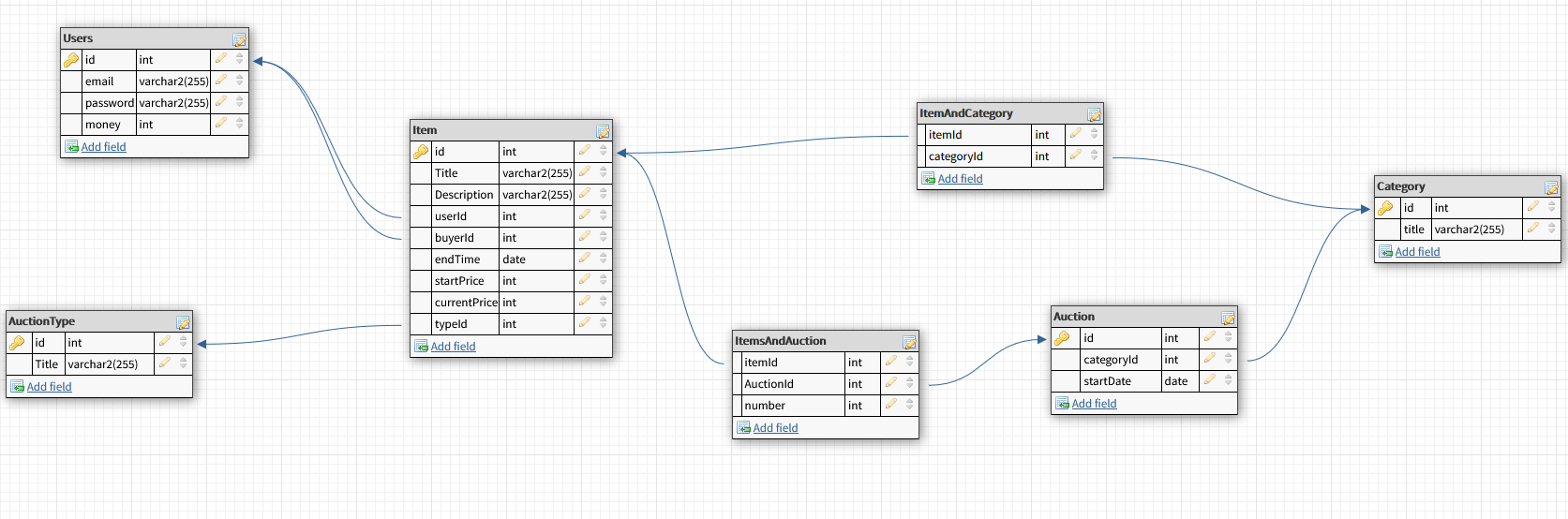


Рисунок 2.1 – Диаграмма базы данных

Таблица users содержит минимальную информацию о пользователях для их авторизации, а именно: идентификатор (id, int, pk), почта (email, varchar2(255), unique), пароль (password, varchar2(255), not null, хешируется), состояние счета (Money, int).

Таблица Item содержит информацию о товарах, а именно: идентификатор товара (id, int, pk), Название товара (Title, varchar2(255)), описание (Description varchar2(255)), идентификатор продавца (userId int fk), идентификатор покупателя(buyerId int fk), время, до которого товар активен (endTime date), начальная цена (startPrice, int), текущая цена (currentPrice, int), тип аукциона (typeId, int fk).

Таблица AuctionType содержит список из двух значений – видов аукционов: динамический и статический. В зависимости от типа аукциона товар может участвовать в разных аукционах. Содержит 2 поля: идентификатор типа аукциона (id, int), название (Title, varchar2(255)).

Таблица ItemAndCategory содержит информацию о том, к каким категориям относится товар, тоесть связь таблицы Category и Item, содержит 2 поля: идентификатор предмета (itemId, int, fk), идентификатор категории (categoryId int).

Таблица Category содержит список категорий в приложении, а именно: идентификатор категории (id, int), название категории (title, varchar2(255)).

Таблица Auction содержит информацию о проходящих динамических аукционах и имеет следующие колонки: идентификатор аукциона (id, int, pk), идентификатор категории(categoryId, int, fk), дата проведения (startDate, date).

Таблица ItemsAndAuction содержит связь предметов с динамическими аукционами, и содержит поля: идентификатор предмета (itemId, int, fk), идентификатор аукциона (auctionId, int, fk), очередь предмета на аукционе (count, int).

## Процедуры для решения поставленных задач

Всего для управления БД было создано 25 процедур. Большинство из них имеют out параметр типа int, который показывает результат выполнения процедуры: 0 – выполнена успешно, -1 – возникли проблемы.

Процедура Insert\_User добавляет новую запись в таблицу Users и имеет следующие параметры: почта (emailp, входящий, varchar2(255), захешированный пароль (passwordp, входящий, varchar2(255)), результат выполнения (resultId, выходящий, int).

Процедура Authorization\_User проверяет вводимый логин и пароль на совпадения в бд и вовзращает 0 в случае успеха и -1 в случае несовпадения данных. Параметры: почта (emailp, входящий, varchar2(255)), пароль (passwordp, входящий, varchar2(255)), результат выполнения (resultId, выходящий, int).

Процедура Registration\_User проверяет валидность почты, и в случае успеха хеширует пароль и вызывает Insert\_User. Параметры: почта (email, входящий, varchar2(255), пароль (password, входящий, varchar2(255)), результат выполнения (resultId, выходящий, int).

Процедура exist\_User проверяет существование пользователя по заданной почте. Параметры: почта (email, входящий, varchar2(255), результат выполнения (resultId, выходящий, boolean).

Процедура Delete\_User проверяет возможность удаления пользователя и в случае успеха удаляет пользователя. Параметры: идентификатор пользователя (idp, входящий, int), результат выполнения (resultId, выходящий, int).

Процедура Addmoney добавляет денег пользователю. Параметры: идентификатор пользователя(idp, входящий, int), сумма денег(valuep, входящий, int).

Процедура Actualize\_item проверяет, продан ли предмет, и в случае успеха переводит деньги на счет продавца, а так же изменяет поля item таким образом, чтобы в приложении он идентифицировался как проданный. Вовзращает 0 в случае успеха и 1 в случае, если не удалось обновить предмет. Параметры: идентификатор предмета (itemidp, входящий, int), результат выполнения(resultId, выходящий, int).

Процедура Get\_Item ищет и выводит первые 100 товаров, которые содержат в описании или названии заданный текст. Параметры: текст для поиска(text, входящий, varchar2).

Процедура Get\_Item\_Category ищет и выводит первые 100 товаров в заданной категории. Параметры: идентификатор категории(typ, входящий, int).

Процедура Insert\_Item добавляет новую запись в таблицу Item. Вовзращает 0 в случае успеха и -1 в случае неудачи. Параметры: название предмета (titlep, входящий, varchar2), описание предмета (description, входящий, varchar2), идентификатор продавца(useridp, входящий, int), время до которого предмет может быть куплен(endTimep, входящий, date), начальная цена(startPricep, входящий, int), тип аукциона, на который выкладывается товар(typeIdp, входящий, int), результат выполнения(resultId, выходящий, int).

Процедура Update\_Price повышает цену на заданный лот на 0.1 от первоначальной стоимости; максимально возможная цена – 2.5 от начальной стоимости. Вовзращает 0 в случае успеха и -1 в случае неудачи. Параметры: идентификатор предмета (idp, входящий, int), идентификатор покупателя(buerIdp, входящий, int), результат выполнения(resultId, выходящий, int).

Процедура Delete\_item удаляет заданный предмет из таблицы, если это возможно. Вовзращает 0, если удаление произошло, в противном случае вовзращает 1. Параметры: идентификатор предмета (idp, входящий, int), результат выполнения(ResultId, выходящий, int).

Процедура Insert\_ItemAndCategory вставляет новую запись в соответствующую таблицу, связывая предмет с категорией. Параметры: идентификатор предмета(itemIdp, входящий, int), идентификатор категории(categoryIdp, входящий, int).

Процедура Delete\_ItemAndCategory удаляет все записи из соответствующей таблицы, где идентификатор предмета равен входному параметру. Параметры: идентификатор предмета(itemIdp, входящий, int).

Процедура Insert\_auction Добавляет новую запись в соответствующую таблицу. В случае успеха вовзращает 0, в противном случае -1. Параметры: Категория аукциона(categoryIdp, входящий, int), дата проведения(startDatep, входящий, date), результат выполнения(ResultId, выходящий, int).

Процедура Delete\_auction удаляет заданный аукцион, если возможно, и очищает связи с ним в таблице ItemsAndAuction. Вовзращает 0 в случае успеха и -1 в случае неудачи. Параметры: идентификатор аукциона(idp, входящий, int), результат выполнения(ResultId, выходящий, int).

Процедура Insert\_ItemsAndAuction добавляет запись в соответствующую таблицу. Параметры: идентификатор предмета(itemIdp, входящий, int), идентификатор аукциона(auctionIdp, входящий, int), очередь товара на заданном аукционе(num, входящий, int).

Процедура Get\_Current вовзращает текущий активный предмет на динамическом аукционе с заданной дату.(Предмет под номером 1 будет активным в период времени с 0:00 до 2:00, 2 – с 2:00-4:00 и т.д.) Параметры: Дата аукциона(datet, входящий, date).

Процедура CleanAuction очищает связи заданного предмета/аукциона в таблице ItemsAndAuction. Если идентификатор передаваемого предмета равен нулю, то очищаются связи заданного аукциона; в противном случае очищаются связи заданного предмета. Параметры: идентификатор предмета (itemIdp, входящий, int), идентификатор аукциона(auctionIdp, входящий, int).

Процедура Fill\_auction заполняет заданный динамический аукцион предметами, подходящими под параметры аукциона(дата, категория). Параметры: идентификатор аукциона (auctionIdp, входящий, int), дата заданного аукциона(cou, входящий, date), идентификатор категории(cat, входящий, int),

Процедура encryptPassword хеширует заданную строку. Параметры: строка для хеширования(casualPassword, выходящий, varchar2).

Процедура export\_users производит экспорт таблицы users в файл по пути c:/xmlFiles/users.txt в формате xml. не имеет параметров.

Процедура import\_users производит импорт данных из файла c:/xmlFiles/users.txt из формата xml в таблицу users. Не имеет параметров.

Процедура ALotItemInsers производит вставку заданного количества записей в таблицу item. Параметры: количество записей(coun входящий int), первая часть названия предметов(strin, входящий, varchar2)(вторая часть добавляется в процессе создания и является порядковым номером записи в итерации),

## Индекс базы данных

Индексы Oracle обеспечивают быстрый доступ к строкам таблиц, сохраняя отсортированные значения указанных столбцов и используя эти отсортированные значения для быстрого нахождения ассоциированных строк таблицы. Индексы позволяют находить строку с определенным значением столбца, просматривая при этом лишь небольшую часть общего объема строк таблицы. Таким образом правильное использование индексов сокращает до минимума количество дорогостоящих операций ввода-вывода.

В данном курсовом проекте также были созданы индексы для каждой таблицы, относящиеся к первичному ключу, а так же индексы для таблиц item(по полю description вместе с title, для быстрого поиска заданной строки) и itemAndCategory(два индекса для полей itemid и categoryid).

## Реализация задач



## Регистрация и авторизация пользователей

Ниже приведены процедуры, реализующие регистрацию и авторизацию пользователей в базе данных.

**Procedure Authorization\_user(**

**emailP VARCHAR,**

**Passwordp Varchar,**

**resultId OUT INTEGER)**

**Is**

**Cou Int;**

**Pw Varchar(255);**

**text varchar(255);**

**Begin**

**Select Count(1) Into cou From Users Where email = emailp;**

**If Cou != 0 Then**

**Select Password Into Pw From Users Where Email=Emailp;**

**text:=passwordp;**

**Encryptpassword(text);**

**if pw=text then**

**Resultid:=0;**

**Else**

**resultId:=-1;**

**End If;**

**end if;**

**End;**

**Procedure Registration\_user(**

**emailP VARCHAR,**

**Passwordp Varchar,**

**resultId OUT INTEGER)**

**Is**

**Cou Int;**

**text varchar(255);**

**Begin**

**if Regexp\_Like (emailP, '[A-z0-9.\_%-]+@[A-z0-9.\_%-]+.[A-z]{2,4}')**

**Then**

**Resultid:=0;**

**Text:=Passwordp;**

**Encryptpassword(Text);**

**Insert\_User(emailp,text , resultId);**

**Else**

**resultId:=-1;**

**end if;**

**End;**

## Добавление новых учетных записей пользователя

Ниже приведена процедура, реализующая добавление учетной записи в таблицу. Так же добавление происходит при помощи процедуры Registration\_user.

**Insert\_User(**

**emailP VARCHAR,**

**Passwordp Varchar,**

**resultId OUT INTEGER)**

**Is**

**cou Int;**

**Begin**

**Select Count(1) Into cou From Users Where email = emailp;**

**IF cou != 0 THEN**

**resultId := -1;**

**ELSE**

**INSERT INTO Users**

**(**

**Email,**

**Password**

**)**

**VALUES**

**(**

**emailP,**

**Passwordp**

**);**

**resultId:=0;**

**End If;**

**End;**

## Добавление лотов

Добавление лотов реализовано в процедуре Insert\_Item.

**Procedure Insert\_Item(**

**Titlep Varchar,**

**Descriptionp Varchar,**

**Useridp Int,**

**Endtimep Date,**

**Startpricep Int,**

**Typeidp Int,**

**Resultid Out Integer)**

**IS**

**cou Int;**

**Begin**

**Select Count(1) Into Cou From Auctiontype Where Id = Typeidp;**

**IF (cou = 0) or (startPricep <=0) THEN**

**resultId := -1;**

**Else**

**INSERT INTO Item**

**(**

**Title,**

**Description,**

**Userid,**

**endTime,**

**Startprice,**

**typeId**

**)**

**VALUES**

**(**

**Titlep,**

**Descriptionp,**

**Useridp,**

**Endtimep,**

**Startpricep,**

**Typeidp**

**);**

**resultId:=0;**

**End If;**

**End;**

* 1. Возможность делать ставки на существующие лоты

Можно делать ставку на существующий лот с помощью процедуры Update\_price.

**Create Or Replace Procedure Updateprice(**

**idp int,**

**Buyeridp Int,**

**Resultid Out Integer)**

**IS**

**Cou Int;**

**Perc Int;**

**couBefore int;**

**bId Int;**

**Sid Int;**

**Mon Int;**

**dat date;**

**Begin**

**Select Currentprice, Startprice,Userid, Buyerid, Currentprice, Endtime Into Cou, Perc, Sid, Bid, Coubefore, dat From Item Where Id = Idp;**

**Select Money Into Mon From Users Where Id=buyerIdp;**

**If ((Cou >= (Perc\*0.1)\*25) or (sId = idp) or (bId=buyeridp)or(mon<cou) or (mon<Perc) or (dat<TRUNC(sysdate)))**

**Then**

**Resultid := -1;**

**Else**

**If Cou = 0**

**Then Cou := Perc;**

**Else Cou := Cou+(Perc\*0.1);**

**End If;**

**Update Users**

**Set Money = Money - Cou**

**where id=buyerIdp;**

**Update Users**

**Set Money = Money + Coubefore**

**where id=bId;**

**Update Item**

**Set Buyerid = Buyeridp,**

**currentPrice = Cou**

**Where Id =Idp;**

**resultId:=0;**

**end if;**

**End;**

## Импорт и экспорт данных.

Для реализации процедур импорта и экспорта потребовалось создать объект директории, которая бы указывала путь для xml файла, в который будет производиться импорт и экспорт. В нашем случае имя директории – «EXPORTFILE», путь – «C:\\xmlFiles». Таблица, над которой будем производить операции – users, файл для записи – users.txt.

Процедура экспорта данных:

**CREATE OR REPLACE PROCEDURE export\_users**

**IS**

**file1 utl\_file.file\_type;**

**xrow CLOB;**

**Begin**

**file1 := UTL\_FILE.FOPEN('EXPORTFILE','users.txt','w');**

**SELECT XMLELEMENT(root,XMLAGG(XMLELEMENT(appuser,**

**XMLATTRIBUTES(**

**E.Id,**

**e.email,**

**e.password,**

**e.Money)**

**))).getCLOBVal() AS xmlsads**

**Into Xrow**

**FROM Users e;**

**utl\_file.put(file1,xrow);**

**utl\_file.fclose(file1);**

**END;**

Процедура импорта из xml файла:

**import\_users**

**IS**

**file1 utl\_file.file\_type;**

**xrow CLOB;**

**BEGIN**

**file1 := UTL\_FILE.FOPEN('EXPORTFILE','users.txt','r');**

**utl\_file.get\_line(file1,xrow);**

**merge INTO Users cur\_t USING**

**(Select Extractvalue(Value(T),'//@ID') Id,**

**extractValue(value(t),'//@EMAIL') email,**

**Extractvalue(Value(T),'//@PASSWORD') Password,**

**extractValue(value(t),'//@MONEY') Money**

**FROM TABLE(XMLSequence(XMLType(xrow).extract('//APPUSER'))) t**

**) imp\_t ON (cur\_t.id=imp\_t.id)**

**WHEN NOT matched THEN**

**INSERT**

**(**

**Cur\_T.Id,**

**Cur\_T.Email,**

**cur\_t.password,**

**cur\_t.Money**

**)**

**VALUES**

**(**

**imp\_t.id,**

**Imp\_T.Email,**

**Imp\_T.Password,**

**imp\_t.Money**

**);**

**Utl\_File.Fclose(File1);**

**End;**

## Описание технологии

Технология резервное копирование и восстановление подразумевает под собой возможность создания бэкапов и вовзрат к ним в случае поломки базы данных. Утилита rman была выбрана мной потому, что по отношению к традиционным базовым возможностям резервирования и восстановления в Oracle, у программы RMAN есть некоторые преимущества, делающие ее в некоторых ситуациях (например, при больших объемах данных) практически незаменимой.

Возможности RMAN включают следующее:

- выполнение полного резервирования и резервирования изменений

- выполнение холодного/горячего резервирования, причем во втором случае табличные пространства не переводятся в режим backup, что позволяет избежать дополнительной нагрузки на журнал

- обнаружение поврежденных блоков

- параллельное выполнения операций ввода/вывода

- автоматическое протоколирование операций копирования и восстановления

Для того, чтобы запустить утилиту rman и подключиться к нашей бд, нужно написать соответствующие команды:

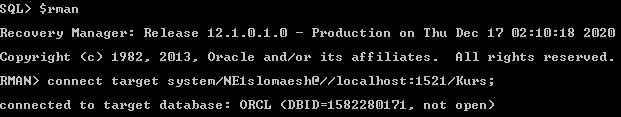


Рисунок 5.1 – запуск утилиты rman

Обратим внимание, что для успешного копирования бд в процессе работы необходимо включить ARCHIVELOG.

Так как подключение произошло напрямую к нашей pdb, мы можем с ней работать напрямую. Для того, чтобы создать backup-файлы, нам необходимо ввести ‘backup database;’, после чего просмотрим созданные файлы с помощью команды ‘list backup;‘.



Рисунок 5.2 – результат работы list backup

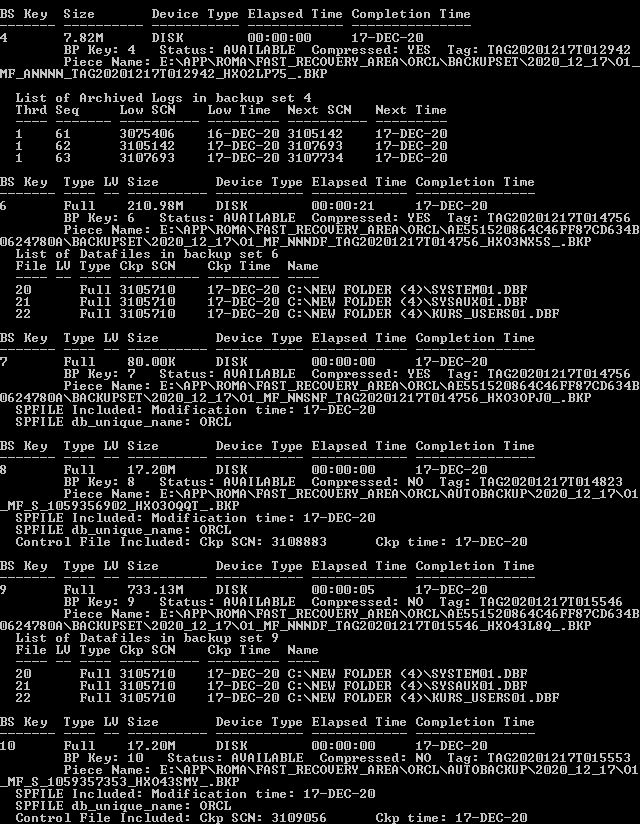


Рисунок 5.3 – результат работы list backup;

Из данного списка мы видим те файлы, которые были созданы с помощью backup database, их размер, местоположение, дату. Для того, чтобы восстановить бд в случае ее поломки, необходимо ввести ‘restore database’. Представим ситуацию, что мы случайно и безвовзратно удалим данные файлы бд:

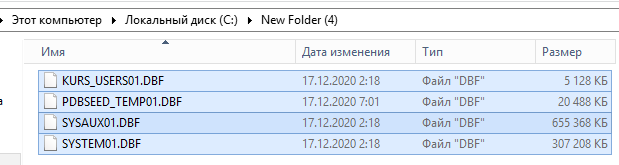


Рисунок 5.4 – файлы для удаления

При последующей попытке запуска бд будет выведена ошибка о том, что система не может найти данные файлы:

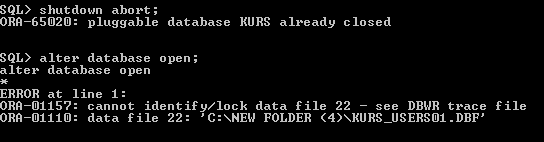


Рисунок 5.5 – неудачная попытка запуска

Теперь нам необходимо восстановить систему до того момента, когда все работало. Для этого нам необходимо снова запустить утилиту rman и подключиться к нашей бд. Далее необходимо запустить команду ‘restore database;’:

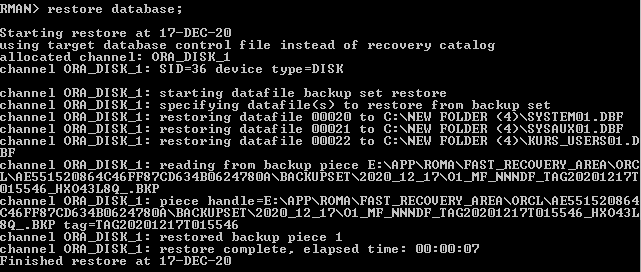


Рисунок 5.6 – восстановление системы

Здесь мы можем видеть, какие файлы были восстановлены, а так же откуда они были восстановлены. Теперь при попытке запустить бд никаких ошибок не будет, и мы сможем спать спокойно:



Рисунок 5.7 – успешный перевод в состояние open.

## Тестирование

Для того, чтобы в процессе работы не возникало ситуаций, когда из-за неправильно введенных данных бд работала некорректно, был введен ряд проверок на валидность данных в процедурах. Большинство процедур имеют вовзращаемый код, значение которого сигнализирует о том, что произошло в процедуре. Значение 0 сигнализирует о том, что все проверки были пройдены успешно и выполнение процедуры не скажется негативным образом на бд; значение -1 обозначает то, что какая-то из проверок не была пройдена и стоит перепроверить входные параметры или условие выполнения процедуры. Первый пример – процедура Registration\_user. В ней введена проверка вводимого адреса электронной почты на соответствие регулярному выражению, дабы в случае опечатки регистрация не была завершена.



Рисунок 6.1 – проверка в функции Registration\_user

Так же может возникнуть ситуация, когда мы захотим повысить ставку на предмет, который уже не продается, т.е. дата продажи уже прошла. Так же во избежание лишней траты денег пользователь не сможет повышать цену, если он был последним, кто сделал ставку или если он является продавцом товара. Так же цена не должна становиться выше максимальной(вспомним, что цена не должна превышать 250% от начальной стоимости). На эти случаи в функции Updateprice предусмотрены проверки на соответствие даты и пользователя и цены:



Рисунок 6.2 – проверка в функции Updateprice

Так же, для защиты паролей пользователей в случае кражи бд было предусмотрено хеширование в базе данных; оно не позволит видеть злоумышленникам исходные данные. За хеширование строки отвечает функция encryptPassword. В ней используется встроенный пакет DBMS\_CRYPTO и функция encrypt. Использование данной процедуры предусмотрено в таких процедурах, как Authorization\_user и Registration\_user.

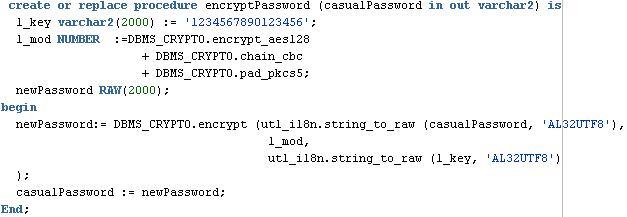


Рисунок 6.3 – вид функции encryptPassword

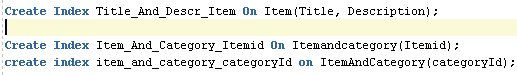
Так же для обеспечения быстродействия поиска в таблицах были добавлены индексы на те поля, которые могут вызвать затруднение. Один из требовательных запросов, требующий время – поиск товара по названию или описанию. Для него, а так же для таблицы ItemToCategory были созданы индексы для ускорения поиска: 

Рисунок 6.4 ­– индексы в базе данных.

## Заключение

В ходе курсовой работы основные задачи была спроектирована база данных для аукциона на базе СУБД Oracle 12c. Основные требования, которые были поставлены к работе, были выполнены:

* Был реализован ряд процедур, соответствующих заявленным функциональным требованиям, а также создающим логическую завершенность;
* Возможность авторизации/регистрации пользователей;
* Для ускорения поиска были реализованы индексы по тем полям, которые могут вызвать задержки в работе процедуры;
* Для защиты пользователей от кражи паролей было введено хеширование пароля на уровне регистрации;
* Реализована технология резервного копирования и восстановления на базе утилиты rman, с помощью которой можно восстановить базу данных даже в случае ее удаления;
* Для исключения ошибочных ситуаций при изменении таблиц была введена валидация данных в большинстве функций, а так же вовзращаемое значение, которое говорит о успешности выполнения процедуры.
* Был реализован экспорт/импорт таблицы users в формат xml.

# Список использованных источников

[1] Oracle-dba.ru [Электронный ресурс] / Режим доступа: https://oracle-dba.ru

[2] stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com

[3] Шифрование в PL/SQL: уроки программирования [Электронный ресурс] / Режим доступа: <https://oracle-patches.com/db/sql/3939-shifrovanie-v-pl-sql-uroki-programmirovaniya>

[4] Developing and Using Stored Procedures [Электронный ресурс] / Режим доступа: <https://docs.oracle.com/cd/B28359_01/appdev.111/b28843/tdddg_procedures.htm>

# ПРИЛОЖЕНИЕ A

Создание базы данных, добавление внешних ключей

*-----------------------------------------------------------  
CREATE TABLE Users*

*(*

*id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,*

*email VARCHAR2(255) UNIQUE NOT NULL,*

*Password Varchar2(255) Not Null,*

*Money Int Default 0);*

*/*

*CREATE TABLE Item (*

*id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,*

*Title VARCHAR2(255) NOT NULL,*

*Description VARCHAR2(255) NOT NULL,*

*userId INT NOT NULL,*

*endTime DATE,*

*startPrice INT NOT NULL,*

*currentPrice INT DEFAULT 0,*

*typeId INT NOT NULL);*

*Alter Table Item*

*add buyerId int default NULL;*

*Alter Table Users*

*Modify Id Generated By Default As Identity;*

*ALTER TABLE Item ADD CONSTRAINT Item\_fk2 FOREIGN KEY (buyerId) REFERENCES Users(id);*

*/*

*CREATE TABLE Category (*

*id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,*

*title VARCHAR2(255) NOT NULL);*

*/*

*CREATE TABLE ItemAndCategory (*

*itemId INT NOT NULL,*

*categoryId INT NOT NULL);*

*alter table ItemAndCategory add constraint constr unique(itemId, categoryId);*

*/*

*Create Table Auction (*

*id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,*

*categoryId INT,*

*startDate DATE NOT NULL);*

*/*

*CREATE TABLE AuctionType (*

*id INT GENERATED ALWAYS AS IDENTITY PRIMARY KEY,*

*Title VARCHAR2(255) NOT NULL);*

*/*

*CREATE TABLE ItemsAndAuction (*

*itemId INT NOT NULL,*

*Auctionid Int Not Null,*

*count INT NOT NULL);*

*/*

*ALTER TABLE Item ADD CONSTRAINT Item\_fk0 FOREIGN KEY (userId) REFERENCES Users(id);*

*ALTER TABLE Item ADD CONSTRAINT Item\_fk1 FOREIGN KEY (typeId) REFERENCES AuctionType(id);*

*ALTER TABLE ItemAndCategory ADD CONSTRAINT ItemAndCategory\_fk0 FOREIGN KEY (itemId) REFERENCES Item(id);*

*ALTER TABLE ItemAndCategory ADD CONSTRAINT ItemAndCategory\_fk1 FOREIGN KEY (categoryId) REFERENCES Category(id);*

*ALTER TABLE Auction ADD CONSTRAINT Auction\_fk0 FOREIGN KEY (categoryId) REFERENCES Category(id);*

*ALTER TABLE ItemsAndAuction ADD CONSTRAINT ItemsAndAuction\_fk0 FOREIGN KEY (itemId) REFERENCES Item(id);*

*ALTER TABLE ItemsAndAuction ADD CONSTRAINT ItemsAndAuction\_fk1 FOREIGN KEY (AuctionId) REFERENCES Auction(id);*

*-----------------------------------------------------------*

# ПРИЛОЖЕНИЕ B

Заранее добавленные данные и процедура добавления n-го кол-ва строк

*-----------------------------------------------------------  
Insert Into Auctiontype*

*(Title)*

*Values*

*('Dynamic');*

*Insert Into Auctiontype*

*(Title)*

*Values*

*('Static');*

*Insert Into Category*

*(Title)*

*Values*

*('Cars');*

*Insert Into Category*

*(Title)*

*Values*

*('Tech');*

*Insert Into Category*

*(Title)*

*Values*

*('art');*

*Insert Into Category*

*(Title)*

*Values*

*('Other');*

*Insert Into Category*

*(Title)*

*Values*

*('Clothes');*

*Create Or Replace Procedure Alotiteminsert(coun int, strin varchar, userId int)*

*Is*

*I Int:=10;*

*cou int;*

*Begin*

*While(I<coun)*

*Loop*

*Insert\_Item(Strin||I,'rand descr'||I, Userid, To\_Date('20210101', 'YYYYMMDD'),I\*10,Mod(I,2)+1,Cou);*

*Select Id Into Cou From Item Where Title = Strin||I;*

*Insert\_Itemandcategory(cou, mod(i, 5)+1);*

*i:=i+1;*

*End Loop;*

*end;*  
*-----------------------------------------------------------*

# ПРИЛОЖЕНИЕ C

Создание процедур

*-----------------------------------------------------------  
--------Users*

*Create Or Replace Procedure Insert\_User(*

*emailP VARCHAR,*

*Passwordp Varchar,*

*resultId OUT INTEGER)*

*Is*

*cou Int;*

*Begin*

*Select Count(1) Into cou From Users Where email = emailp;*

*IF cou != 0 THEN*

*resultId := -1;*

*ELSE*

*INSERT INTO Users*

*(*

*Email,*

*Password*

*)*

*VALUES*

*(*

*emailP,*

*Passwordp*

*);*

*resultId:=0;*

*End If;*

*End;*

*/*

*Create Or Replace Procedure Authorization\_user(*

*emailP VARCHAR,*

*Passwordp Varchar,*

*resultId OUT INTEGER)*

*Is*

*Cou Int;*

*Pw Varchar(255);*

*text varchar(255);*

*Begin*

*Select Count(1) Into cou From Users Where email = emailp;*

*If Cou != 0 Then*

*Select Password Into Pw From Users Where Email=Emailp;*

*text:=passwordp;*

*Encryptpassword(text);*

*if pw=text then*

*Resultid:=0;*

*Else*

*resultId:=-1;*

*End If;*

*end if;*

*End;*

*Create Or Replace Procedure Registration\_user(*

*emailP VARCHAR,*

*Passwordp Varchar,*

*resultId OUT INTEGER)*

*Is*

*Cou Int;*

*text varchar(255);*

*Begin*

*if Regexp\_Like (emailP, '[A-z0-9.\_%-]+@[A-z0-9.\_%-]+.[A-z]{2,4}')*

*Then*

*Resultid:=0;*

*Text:=Passwordp;*

*Encryptpassword(Text);*

*Insert\_User(emailp,text , resultId);*

*Else*

*resultId:=-1;*

*end if;*

*End;*

*/*

*CREATE OR REPLACE PROCEDURE exist\_user(*

*emailP VARCHAR,*

*resultId OUT BOOLEAN)*

*IS*

*cou int;*

*Begin*

*Select Count(1) Into Cou From Users Where email = Emailp;*

*IF cou != 0 THEN*

*resultId:=true;*

*ELSE*

*resultId := false;*

*End If;*

*END;*

*/*

*Create Or Replace Procedure Delete\_User(--проверить каскадно*

*idp int,*

*Resultid Out int)*

*Is*

*cou int;*

*Begin*

*Select Count(1) Into Cou From Item Where Userid=Idp;*

*If Cou = 0 Then*

*Delete From Users Where Users.Id = Idp;*

*resultId:=0;*

*Else*

*resultId:=-1;*

*end if;*

*End;*

*/*

*Create Or Replace Procedure Addmoney(*

*Idp Int,*

*Valuep Int)*

*Is*

*Begin*

*Update Users*

*Set money = money + valuep*

*Where Id =Idp;*

*end;*

*/*

*--------Item*

*/*

*Create Or Replace Procedure Actualize\_Item(*

*Itemidp Int,*

*resultId out int)*

*Is*

*Cou Int;*

*Datep Date;*

*Price Int;*

*Begin*

*Select Userid, Endtime, Currentprice Into Cou, Datep, Price From Item Where Id = Itemidp;*

*If Datep < Trunc(Sysdate) And Price != 0 Then*

*Update Users*

*Set Money = Money + Price*

*Where Id = cou;*

*Update Item*

*Set Currentprice = 0, endTime = TRUNC(sysdate-1)*

*Where Id = Itemidp;*

*Resultid:=0;*

*Else*

*Resultid:=-1;*

*end if;*

*end;*

*create or replace*

*Procedure Get\_Item(*

*text varchar*

*)*

*Is*

*Cursor Cur Is Select id, Title, Description, Endtime, Startprice, Currentprice From Item Where*

*( Title Like '%' || Text || '%' Or Description Like '%' || Text || '%') And Typeid = 2;*

*rec cur%rowtype;*

*Begin*

*for rec in cur*

*Loop*

*If(Cur%Rowcount<101) Then*

*Dbms\_Output.Put\_Line(Cur%Rowcount||' '||Rec.id||' '||Rec.Title||' '||*

*Rec.Description||' '||Rec.Endtime|| ' '||*

*Rec.Startprice||' '||Rec.Currentprice);*

*Else Exit;*

*End If;*

*End Loop;*

*--close cur;*

*end;*

*Create Or Replace*

*Procedure Get\_Item\_Category(*

*typ int*

*)*

*Is*

*Cursor Cur Is Select item.id, Item.Title, Item.Description, Item.Endtime, Item.Startprice, Item.Currentprice, Category.Title cat\_title From Item Join Itemandcategory Iac On Item.Id = Iac.Itemid Join Category On Iac.Categoryid = Category.Id Where*

*category.id = typ And Typeid = 2;*

*rec cur%rowtype;*

*Begin*

*for rec in cur*

*Loop*

*If(Cur%Rowcount<101) Then*

*Dbms\_Output.Put\_Line(Cur%Rowcount||' '||Rec.id||' '||Rec.Title||' '||*

*Rec.Description||' '||Rec.Endtime|| ' '||*

*Rec.Startprice||' '||Rec.Currentprice||' '||Rec.Cat\_Title);*

*else exit;*

*end if;*

*End Loop;*

*--close cur;*

*end;*

*Create Or Replace Procedure Insert\_Item(*

*Titlep Varchar,*

*Descriptionp Varchar,*

*Useridp Int,*

*Endtimep Date,*

*Startpricep Int,*

*Typeidp Int,*

*Resultid Out Integer)*

*IS*

*cou Int;*

*Begin*

*Select Count(1) Into Cou From Auctiontype Where Id = Typeidp;*

*IF (cou = 0) or (startPricep <=0) THEN*

*resultId := -1;*

*Else*

*INSERT INTO Item*

*(*

*Title,*

*Description,*

*Userid,*

*endTime,*

*Startprice,*

*typeId*

*)*

*VALUES*

*(*

*Titlep,*

*Descriptionp,*

*Useridp,*

*Endtimep,*

*Startpricep,*

*Typeidp*

*);*

*resultId:=0;*

*End If;*

*End;*

*Create Or Replace Procedure Updateprice(*

*idp int,*

*Buyeridp Int,*

*Resultid Out Integer)*

*IS*

*Cou Int;*

*Perc Int;*

*couBefore int;*

*bId Int;*

*Sid Int;*

*Mon Int;*

*dat date;*

*Begin*

*Select Currentprice, Startprice,Userid, Buyerid, Currentprice, Endtime Into Cou, Perc, Sid, Bid, Coubefore, dat From Item Where Id = Idp;*

*Select Money Into Mon From Users Where Id=buyerIdp;*

*If ((Cou >= (Perc\*0.1)\*25) or (sId = idp) or (bId=buyeridp)or(mon<cou) or (mon<Perc) or (dat<TRUNC(sysdate)))*

*Then*

*Resultid := -1;*

*Else*

*If Cou = 0*

*Then Cou := Perc;*

*Else Cou := Cou+(Perc\*0.1);*

*End If;*

*Update Users*

*Set Money = Money - Cou*

*where id=buyerIdp;*

*Update Users*

*Set Money = Money + Coubefore*

*where id=bId;*

*Update Item*

*Set Buyerid = Buyeridp,*

*currentPrice = Cou*

*Where Id =Idp;*

*resultId:=0;*

*end if;*

*End;*

*create or replace*

*Procedure Delete\_Item(*

*Idp Int,*

*resultId out int)*

*Is*

*Cou Int;*

*bid int;*

*Begin*

*Select Count(1) Into Cou From Item Where Id = Idp;*

*If Cou = 0*

*Then*

*Resultid:=-1;*

*Else*

*Select Currentprice Into Bid From Item Where Id = Idp;*

*If Bid != 0*

*Then*

*Resultid:=-1;*

*Else*

*Select Typeid Into Bid From Item Where Id = Idp;*

*if bId = 1 then*

*Cleanauction(Idp, 0);*

*end if;*

*Delete\_Itemandcategory(idp);*

*Delete From Item Where Id=Idp;*

*resultId:=0;*

*End If;*

*end if;*

*End;*

*/*

*--ItemAndCategory*

*/*

*Create Or Replace Procedure Insert\_Itemandcategory(*

*Itemidp Int,*

*Categoryidp Int)*

*Is*

*Begin*

*INSERT INTO ItemAndCategory*

*(*

*Itemid,*

*categoryId*

*)*

*VALUES*

*(*

*Itemidp,*

*categoryIdp*

*);*

*end;*

*Create Or Replace Procedure Delete\_Itemandcategory(*

*Itemidp Int)*

*Is*

*Begin*

*Delete From ItemAndCategory Where Itemid=Itemidp;*

*end;*

*/*

*--auction*

*/*

*Create Or Replace Procedure Insert\_Auction(*

*Categoryidp Int,*

*Startdatep Date,*

*resultId out int)*

*Is*

*cou int;*

*Begin*

*Select count(1) Into Cou From Auction Where Startdate = Startdatep;*

*If Startdatep > Sysdate and cou=0 Then*

*INSERT INTO Auction*

*(*

*Categoryid,*

*startDate*

*)*

*VALUES*

*(*

*Categoryidp,*

*startDatep*

*);*

*resultId := 0;*

*Else*

*Resultid:=-1;*

*end if;*

*end;*

*Create Or Replace Procedure Delete\_Auction(*

*Idp Int,*

*resultId out int)*

*Is*

*Cou Date;*

*tes date := TRUNC(sysdate);*

*bid int;*

*Begin*

*Select Count(1) Into Bid From Auction Where Id = Idp;*

*If Bid=0 Then*

*Resultid:=-1;*

*else*

*Select Startdate Into Cou From Auction Where Id = Idp;*

*If cou <= tes*

*Then*

*Resultid:=-1;*

*Else*

*CleanAuction(0, idP);*

*Delete From Auction Where Id=Idp;*

*resultId:=0;*

*End If;*

*End If;*

*End;*

*/*

*--itemsAndAuction*

*/*

*Create Or Replace Procedure Insert\_Itemsandauction(*

*Itemidp Int,*

*Auctionidp Int,*

*num int)*

*Is*

*cou int;*

*Begin*

*Select Count(1) Into Cou From Itemsandauction Where Auctionid=Auctionidp And Count=Num;*

*If Cou=0 Then*

*Select Count(1) Into Cou From Itemsandauction Where Itemid = Itemidp;*

*if cou=0 then*

*INSERT INTO ItemsAndAuction*

*(*

*Itemid,*

*Auctionid,*

*count*

*)*

*VALUES*

*(*

*Itemidp,*

*Auctionidp,*

*Num);*

*End If;*

*end if;*

*End;*

*Create Or Replace Procedure Get\_Current(*

*Datet date*

*)*

*Is*

*Cursor Cur Is Select Item.Title, Item.Description, Item.Endtime, Item.Startprice, Item.currentPrice From Item join ItemsAndAuction on item.id = ItemsAndAuction.itemId join Auction on ItemsAndAuction.auctionId = Auction.id Where (Itemsandauction.Count = Round(To\_Number(To\_Char(Datet, 'HH24'))/2 + 0.9)) and Auction.startDate = (TRUNC(datet)) ;*

*rec cur%rowtype;*

*Begin*

*for rec in cur*

*loop*

*Dbms\_Output.Put\_Line(Cur%Rowcount||' '||Rec.Title||' '||*

*Rec.Description||' '||Rec.Endtime|| ' '||*

*Rec.Startprice||' '||rec.currentPrice);*

*End Loop;*

*end;*

*create or replace*

*Procedure CleanAuction(*

*Itemidp Int,*

*auctionIdp int)*

*Is*

*Dat Date;*

*Tes Date := Trunc(Sysdate);*

*cou int;*

*Begin*

*If Itemidp=0 Then*

*Select Startdate Into Dat From Auction Where id=Auctionidp;*

*If Dat != Tes Then*

*Delete From Itemsandauction Where Auctionid=Auctionidp;*

*End If;*

*Else*

*Select Count(1) Into Cou From Auction Inner Join Itemsandauction On Id=Itemsandauction.Auctionid And Itemsandauction.Itemid=Itemidp;*

*If Cou=0 Then*

*Cou:=0;*

*else*

*Select Auction.Startdate Into Dat From Auction Inner Join Itemsandauction On Id=Itemsandauction.Auctionid And Itemsandauction.Itemid=Itemidp;*

*If Dat != Tes Then*

*Delete From Itemsandauction Where Itemid=Itemidp;*

*end if;*

*End If;*

*End If;*

*End;*

*Create Or Replace Procedure Fill\_Auction(*

*Auctionidp Int,*

*Cou Date,*

*cat int)*

*Is*

*Cursor Cur Is Select Item.Id From Item Left Outer Join Itemsandauction On Item.Id = Itemsandauction.Itemid Join Itemandcategory On Item.Id = Itemandcategory.Itemid*

*Where Typeid=1 And Endtime>=Cou and categoryId=cat and CurrentPrice=0;*

*rec cur%rowtype;*

*I Int:=1;*

*Begin*

*Open Cur;*

*Fetch Cur Into rec;*

*while(cur%found and i != 13)*

*Loop*

*Insert\_Itemsandauction(rec.id,auctionIdP,i);*

*Fetch Cur Into rec;*

*i:=i+1;*

*End Loop;*

*close cur;*

*end;*

*create or replace procedure encryptPassword (casualPassword in out varchar2) is*

*l\_key varchar2(2000) := '1234567890123456';*

*l\_mod NUMBER :=DBMS\_CRYPTO.encrypt\_aes128*

*+ DBMS\_CRYPTO.chain\_cbc*

*+ DBMS\_CRYPTO.pad\_pkcs5;*

*newPassword RAW(2000);*

*begin*

*newPassword:= DBMS\_CRYPTO.encrypt (utl\_i18n.string\_to\_raw (casualPassword, 'AL32UTF8'),*

*l\_mod,*

*utl\_i18n.string\_to\_raw (l\_key, 'AL32UTF8')*

*);*

*casualPassword := newPassword;*

*End;*

*--XML--*

*CREATE OR REPLACE DIRECTORY EXPORTFILE*

*As*

*'C:\\xmlFiles';*

*CREATE OR REPLACE PROCEDURE export\_users*

*IS*

*file1 utl\_file.file\_type;*

*xrow CLOB;*

*Begin*

*file1 := UTL\_FILE.FOPEN('EXPORTFILE','users.txt','w');*

*SELECT XMLELEMENT(root,XMLAGG(XMLELEMENT(appuser,*

*XMLATTRIBUTES(*

*E.Id,*

*e.email,*

*e.password,*

*e.Money)*

*))).getCLOBVal() AS xmlsads*

*Into Xrow*

*FROM Users e;*

*utl\_file.put(file1,xrow);*

*utl\_file.fclose(file1);*

*END;*

*CREATE OR REPLACE PROCEDURE import\_users*

*IS*

*file1 utl\_file.file\_type;*

*xrow CLOB;*

*BEGIN*

*file1 := UTL\_FILE.FOPEN('EXPORTFILE','users.txt','r');*

*utl\_file.get\_line(file1,xrow);*

*merge INTO Users cur\_t USING*

*(Select Extractvalue(Value(T),'//@ID') Id,*

*extractValue(value(t),'//@EMAIL') email,*

*Extractvalue(Value(T),'//@PASSWORD') Password,*

*extractValue(value(t),'//@MONEY') Money*

*FROM TABLE(XMLSequence(XMLType(xrow).extract('//APPUSER'))) t*

*) imp\_t ON (cur\_t.id=imp\_t.id)*

*WHEN NOT matched THEN*

*INSERT*

*(*

*Cur\_T.Id,*

*Cur\_T.Email,*

*cur\_t.password,*

*cur\_t.Money*

*)*

*VALUES*

*(*

*imp\_t.id,*

*Imp\_T.Email,*

*Imp\_T.Password,*

*imp\_t.Money*

*);*

*Utl\_File.Fclose(File1);*

*End;*