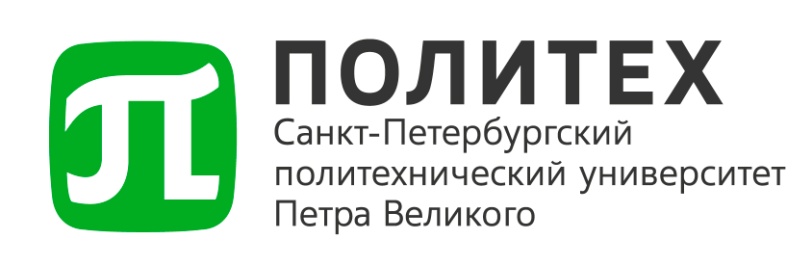
ФЕД­­ЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**



**Курсовая работа**

**«База данных датчиков измерения механических величин»**

**по дисциплине «Управление данными в приборостроении»**

Выполнил

студент гр.23535/2 подпись \_\_\_\_\_\_\_\_\_\_ Летюшев А.И.

Преподаватель подпись \_\_\_\_\_\_\_\_\_\_ Малыхина Г. Ф.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2018 г.

Санкт-Петербург

2018

Оглавление

Введение 1

1. Моделирование данных 2

1.1 E/R-модель данных2

1.2 ODL-модель 3

2. База данных Oracle 6

2.1 Создание базы данных

датчиков измерения механических величин6

2.2 Запросы к базе данных

датчиков измерения механических величин7

3. Триггеры базы данных 14

3.1 Триггер уровня строки, с псевдозаписями 14

3.2 Триггер с предикатами if-then15

3.3 Триггер с условием when 16

4. Программирование клиента базы данных17

4.1 Первая транзакция17

4.2 Вторая транзакция19

Заключение 21

Список источников 22

Введение

Целью данной курсовой работы является изучение систем управления базами данных в локальных и распределенных вычислительных системах, знакомство с моделями представления баз данных, языками моделирования, манипулирования данных и языками запросов, с основными принципами, функциями и организацией баз данных.

В этой работе будут рассмотрены объектно-ориентированное проектирование баз данных, реляционные модели и реляционная алгебра запросов, представлены основные методы и операторы языка запросов SQL, средства управления и администрирования баз данных с помощью функций JDBC.

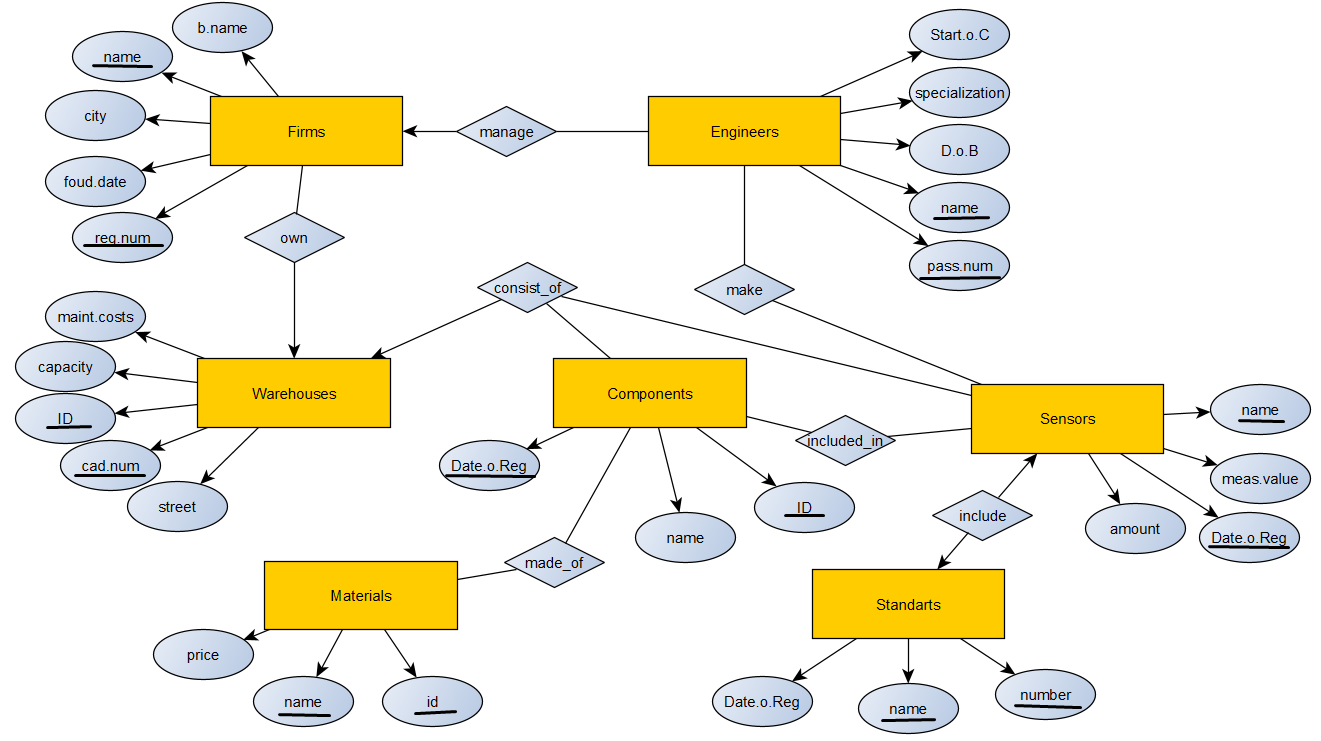
Актуальность проделанной работы состоит в изучении современных средств программирования в СУБД Oracle, получении навыков администрирования баз данных, ознакомлениями с основными концепциями программирования клиент-серверных приложения на прикладных языках программирования, таких как Java и языке запросов SQL.

1. Моделирование данных

Проектирование базы данных начинается с построения моделей данных, разработки их организации и представления. Например, необходимо выполнить описание хранимой информации в виде объектов, содержащих определенный кортежи данных и имеющих уникальные ключевые атрибуты, а также связи с другими объектами. Традиционно выполняют построение E/R-модели или объектно-ориентированной ODL модели.

1.1 E/R-модель данных

Схема E/R-модели содержит сущности, атрибуты и связи. E/R-модель позволяет представить 3 типа связей сущностей: связи один к одному, связь одного объекта со многими и множественные связи между объектами сущностей.



1.2 ODL-модель

Язык ODL предназначен для определения схемы данных в объектно-ориентированной нотации. Это язык представления данных, он не предназначен для выполнения запросов или операций манипулирования данными.

Interface Firms(key name,reg.num)

{

Attribute string b.name;

Attribute string name;

Attribute int reg.num;

Attribute string city;

Attribute int found.date;

Relationship set<Engineers> manage;

Inverse Emgimeers :: work;

Relationship set<Warehouses> own;

Inverse Warehouses :: belongs;

}

Interface Engineers(key name,pass.num)

{

Attribute string name;

Attribute string specialization;

Attribute int D.o.B;

Attribute int pass.num;

Attribute int Start.o.C;

Relationship Firms works;

Inverse Firms :: manage;

Relationship set<Sensors> make;

Inverse Sensors :: made\_by;

}

Interface Sensors(key Date.o.Reg,name)

{

Attribute string name;

Attribute string meas.value;

Attribute int amount;

Attribute int Date.o.Reg;

Relationship set<Engineers> made\_by;;

Inverse Engineers :: make;

Relationship set<Components> consist\_of;

Inverse Components :: included\_in;

Relationship Warehouses stored\_in;

Inverse Warehouses :: consist\_of;

Relationship Standarts belongs;

Inverse Standarts :: include;

}

Interface Components(key ID,Date.o.Reg)

{

Attribute string standard;

Attribute string material;

Attribute int ID;

Attribute int Date.o.Reg;

Relationship set<Sensors> included\_in;

Inverse Sensors :: consist\_of;

Relationship Warehouses stored\_in;

Inverse Warehouses :: consist\_of;

Relationship set<Materials> belongs;

Inverse Materials :: made\_of;

}

Interface Warehouses(key ID,cad.num)

{

Attribute string street;

Attribute int ID;

Attribute int capacity;

Attribute int cad.num;

Attribute int maint.costs;

Relationship set<Sensors> consist\_of;

Inverse Sensors :: stored\_in;

Relationship set<Components> consist\_of;

Inverse Components :: stored\_in;

Relationship Firms belongs;

Inverse Firms own;

}

Interface Materials(key name,ID)

{

Attribute string name;

Attribute int ID;

Attribute int price;

Relationship set<Components> made\_of;

Inverse Components:: belongs;

}

Interface Standarts(key name,number)

{

Attribute string name;

Attribute int number;

Attribute int Date.o.Reg;

Relationship Sensors include;

Inverse Sensors :: belongs;

}

База данных находится в нормальной форме Бойса-Кодда (НФБК), так как все атрибуты являются простыми, все не ключевые атрибуты отношений зависят от ключей, нет транзитивных зависимостей и ключи некоторых отношений являются составными.

2. База данных Oracle

SQL – язык структурированных запросов баз данных. В данной курсовой работе используется стандарт SQL3, содержащие наиболее совершенные средства работы с данными..

2.1 Создание базы данных датчиков измерения механических величин с помощью операции create table и заполнение таблиц методом insert

Оператор CREATE TABLE создает новые таблицы, которые заполняются с помощью INSERT. DROP TABLE удаляет таблицы, с помощью ALTER TABLE таблицы могут быть изменены. SQL-запросы основаны на предложении SELECT-FROM-WHERE, применяемое к одному или объединению отношений.

drop table Firms cascade constraints;

CREATE TABLE Firms (Name\_f varchar2(255) PRIMARY KEY,BossName varchar2(255),City varchar2(255), Reg int,FoundD int);

drop table Engineers cascade constraints;

CREATE TABLE Engineers (Pass int PRIMARY KEY,EngName varchar2(255),Spec varchar2(255),YearOB int,StartOC date,Name\_f varchar2(255) REFERENCES Firms(Name\_f));

drop table Warehouses cascade constraints;

CREATE TABLE Warehouses (WarehID int PRIMARY KEY,Cadastr int,Capacity int,Costs int,St varchar2(255), Name\_f varchar2(255) REFERENCES Firms(Name\_f));

drop table Sensors cascade constraints;

CREATE TABLE Sensors ( SensName varchar2(255) PRIMARY KEY, Mvalue varchar2(255), DoR date, amount int,Pass int REFERENCES Engineers(Pass) );

drop table Stands;

CREATE TABLE Stands ( StandName varchar2(255) PRIMARY KEY, StandNum int,DoR date,SensName varchar2(255) REFERENCES Sensors(SensName));

drop table Comps;

CREATE TABLE Comps (CompID int PRIMARY KEY, CompDoR date,CompName varchar2(255), WarehID int REFERENCES Warehouses(WarehID));

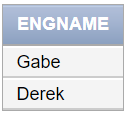
drop table Materials;

CREATE TABLE Materials ( MatName varchar2(255) PRIMARY KEY, mID int,Mprice int, WarehID int REFERENCES Warehouses(WarehID));

2.2 Запросы к базе данных датчиков измерения механических величин

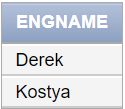
1.Имя инженера у которого специальность IT1:

SELECT EngName FROM Engineers WHERE Spec = 'IT1'



2.Имя инженера который работает на фирму которая находится в Петербурге

SELECT EngName FROM Engineers,Firms WHERE Firms.city = 'S.Peterburg' AND Engineers.Name\_f=Firms.Name\_f



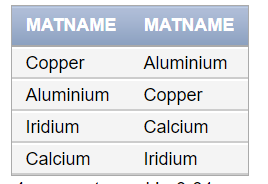
3. Имя работника который работает на фирму где склад находится на улице Nostyushka

SELECT EngName FROM Engineers,Firms,Warehouses WHERE Warehouses.st = 'Nostyushka' AND Engineers.Name\_f=Firms.Name\_f AND Warehouses.Name\_f=Firms.Name\_f



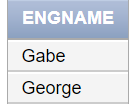
4)Вывести попарно имена Материалов которые находятся на одном складе:

Select DISTINCT d1.MatName,d2.MatName FROM Materials d1,Materials d2 WHERE d1.MatName!=d2.MatName and d1.WarehID=d2.WarehID



5)Вывести имена инженеров работающих в компании где рег.номер 111111:

SELECT EngName FROM Engineers WHERE Name\_f = (SELECT Name\_f FROM Firms WHERE Reg=11111111)



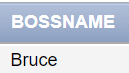
Вывести ID склада где директора зовут Kirill:

SELECT WarehID FROM Warehouses WHERE Name\_f =( SELECT Name\_f FROM Firms WHERE BossName='Kirill')



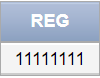
Вывести имя босса фирмы которое владеет складом где находится компонент с id 1:

Select Bossname From Firms WHERE Name\_f=(SELECT Name\_f FROM Warehouses WHERE WarehID = (Select WarehID FROM Comps WHERE CompID=1))



Вывести рег.номер фирмы которая владеет складом где находится материал с id=31:

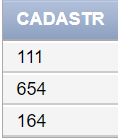
SELECT Reg FROM Firms WHERE Name\_f=(SELECT Name\_f FROM Warehouses WHERE WarehID = (SELECT WarehID FROM Materials WHERE mID=31))



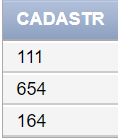
IN/EXISTS:

Вывести кадастровый номер склада который принадлежит фирме в Москве

SELECT Cadastr FROM Warehouses WHERE Name\_f IN(SELECT Name\_f FROM Firms WHERE city='Moscow')



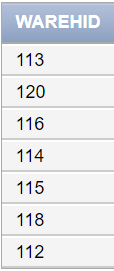
SELECT Cadastr FROM Warehouses WHERE EXISTS( SELECT \* FROM Firms WHERE city='Moscow' AND Firms.Name\_f=Warehouses.Name\_f)



ALL/ANY:

Вывести все склады которые не принадлежат компании в Москве

SELECT WarehID FROM Warehouses WHERE Name\_f <> all( SELECT Name\_f FROM Firms WHERE city='Moscow')

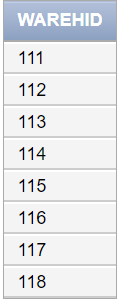


SELECT WarehID FROM Warehouses WHERE NOT Name\_f = ANY( SELECT Name\_f FROM Firms WHERE city='Moscow')

6)

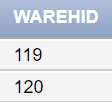
Вывести все склады где есть материалы:

SELECT WarehID FROM Warehouses INTERSECT SELECT WarehID FROM Materials



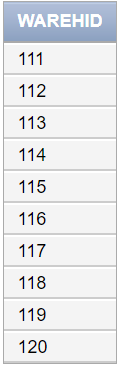
Склады где нет материалов:

SELECT WarehID FROM Warehouses Minus SELECT WarehID FROM Materials



Склады где и есть материалов и нет

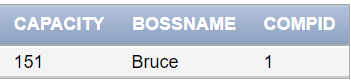
SELECT WarehID FROM Warehouses UNION SELECT WarehID FROM Materials



7)Создать простое пользовательское представление складов содержащее: вместимость склада,имя босса фирмы которая им владеет и где находится CompID 1

CREATE VIEW WFC AS (SELECT capacity,BossName,CompID FROM Warehouses,Firms,Comps WHERE Warehouses.Name\_f = Firms.Name\_f AND Comps.WarehID = Warehouses.WarehID)

SELECT \* FROM WFC WHERE CompID = 1



8)

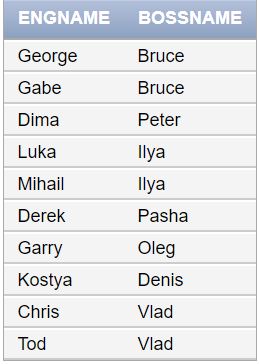
Вывести id склада и имя босса фирмы которая владеет складом:

SELECT BossName,WarehID FROM Warehouses JOIN Firms ON Warehouses.Name\_f = Firms.Name\_f



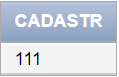
Вывести имена инженеров и имя босса на которого они работают

SELECT EngName,Bossname FROM Engineers CROSS JOIN Firms WHERE Engineers.Name\_f = Firms.Name\_f



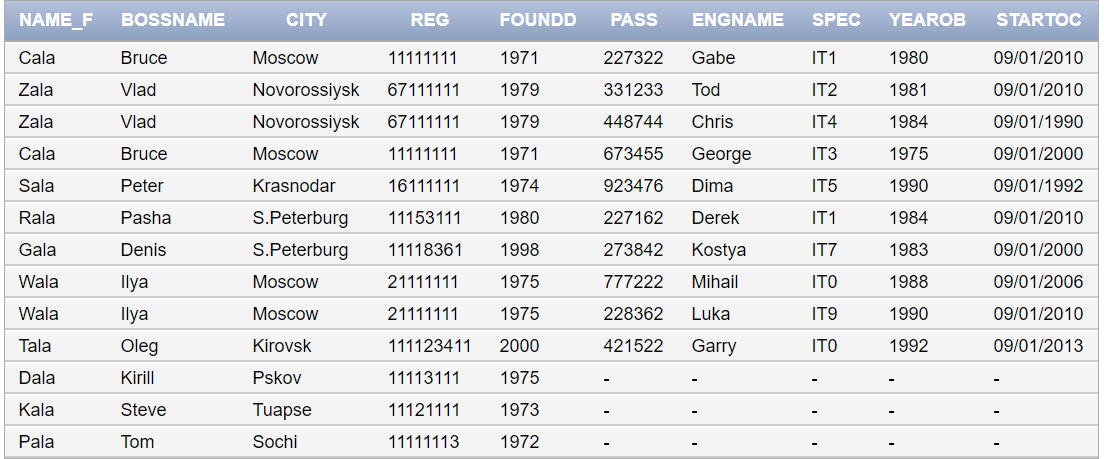
Вывести кадастровый номер склада где находятся компоненты с id 1

SELECT Cadastr FROM Warehouses NATURAL JOIN Comps WHERE CompID = 1



Вывести информацию обо всех фирмах и инженерах, работающих в них, включая фирмы, не имеющих инженеров, и включая инженеров, не работающих в фирмах

SELECT \* FROM Firms NATURAL FULL OUTER JOIN Engineers



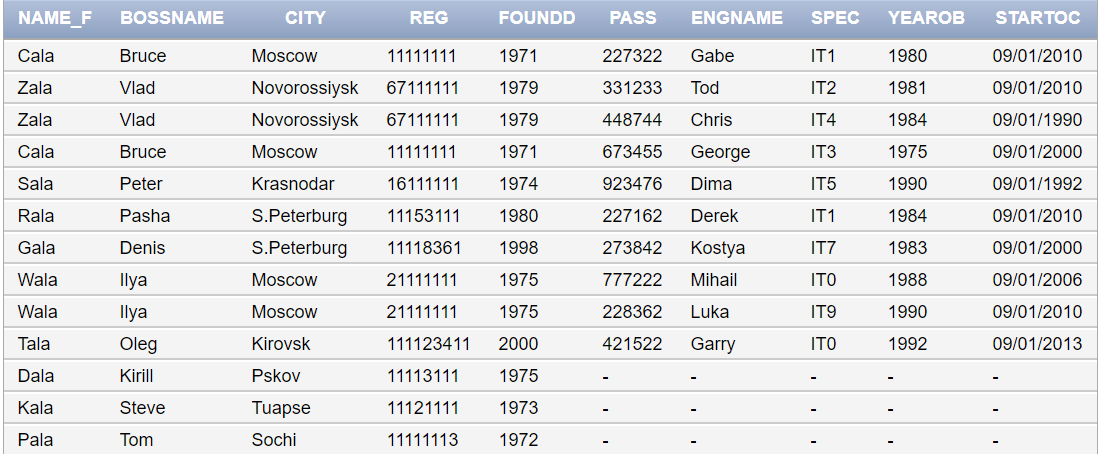
Вывести информацию обо всех фирмах и инженерах, работающих в них, не включая фирмы, не имеющих инженеров, и включая инженеров, не работающих в фирмах

SELECT \* FROM Firms NATURAL RIGHT OUTER JOIN Engineers



Вывести информацию обо всех фирмах и инженерах, работающих в них, включая фирмы, не имеющих инженеров, и не включая инженеров, не работающих в фирмах

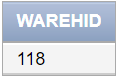
SELECT \* FROM Firms NATURAL LEFT OUTER JOIN Engineers



9)

Вывести id склада где вместимость самая большая

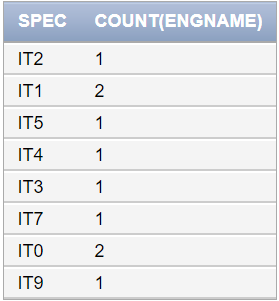
SELECT WarehID FROM Warehouses WHERE capacity=(SELECT MAX(capacity) FROM Warehouses)



10)

сколько инженеров работают на одной специальности

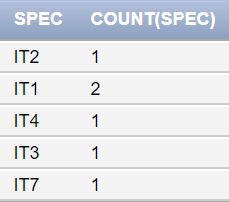
SELECT Spec, COUNT(EngName) FROM Engineers GROUP BY Spec



11)

Сколько разных типов инженеров есть,где хотя бы один родился раньше 1986

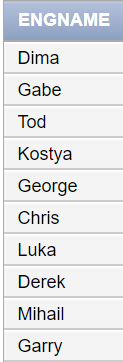
SELECT Spec, Count(Spec) FROM Engineers GROUP BY Spec HAVING MIN(YearOB)<1986



12)

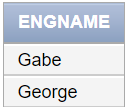
Имена инженеров которые есть

SELECT DISTINCT m1.EngName FROM Engineers m1 WHERE m1.EngName IN(SELECT m2.EngName FROM Engineers m2 WHERE m1.EngName=m2.EngName)



Имена инженеров которые работают на фирму Cala

SELECT EngName FROM Engineers WHERE EXISTS( SELECT \* FROM Firms WHERE Firms.Name\_f=Engineers.Name\_f AND Firms.Name\_f='Cala')



3. Триггеры базы данных

В стандарте SQL3 предусмотрены триггеры – обработчики, выполняющие проверку ограничений. Условия проверки ограничений задаются программно, в ответ на событие производится действие, которое может выполнятся как до наступления, так и вместо или после наступления события.

1. Триггер уровня строки, использующий псевдозаписи new предназначенный для поддержания ссылочной целостности.

При изменениях (обновлении) данных в таблице Sensors триггер обновляет связанные с этой таблицей дочерние записи в таблице Stands, на которые есть ссылка.

create or replace trigger Sensors\_trigger

after update of SensName on Sensors

for each row

begin

update Stands set SensName=:new.SensName where SensName=:old.SensName;

end Sensors\_trigger;

Приведен пример срабатывания триггера при обновлении в таблице Sensors

значения SensName.

select \* from Sensors where SensName='Watch' ;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SENSNAME** | **MVALUE** | **DOR** | **AMOUNT** | **PASS** |
| Watch | sec | 09/03/2005 | 11 | 227162 |

update Sensors set SensName='Watch\_2' where SensName='Watch';

select \* from Sensors where SensName='Watch\_2' ;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SENSNAME** | **MVALUE** | **DOR** | **AMOUNT** | **PASS** |
| Watch\_2 | sec | 09/03/2005 | 11 | 227162 |

select \* from Stands where SensName='Watch\_2' ;

|  |  |  |  |
| --- | --- | --- | --- |
| **STANDNAME** | **STANDNUM** | **DOR** | **SENSNAME** |
| Military standard | 16 | 07/13/2000 | Watch\_2 |

2. Триггер с предикатами IF-THEN для наиболее полного аудита действий пользователя

Создадим новую таблицу аудита материалов, в которую будем записывать все изменения данных (пользователь который вносил изменения (user) , дату совершения операции (time), операцию (operation) и само историю изменения (history).

create table Materials\_audit ( userID varchar2 (30), time Date, operation varchar2 (20), history varchar2(60));

При любой операции с таблицей Materials триггер будет делать запись в таблице аудита.

create or replace trigger Materials\_trigger

after insert or delete or update on Materials

for each row

declare

operation varchar2(50);

begin

if inserting then operation :='Insert'; End if;

if deleting then operation :='Delete'; End if;

if updating then operation :='Update'; End if;

insert into Materials\_audit values (user, sysdate, operation ,

' Old materials: '||:old.MatName||' and Newmaterials: '||:new.MatName);

end Materials\_trigger;

Например добавим таблицу новую запись.

insert into Materials values ( 'Aurum', 55, 700 ,118 );

Таблица аудита зафиксирует совершенную операцию.

select \* from Materials\_audit ;

|  |  |  |  |
| --- | --- | --- | --- |
| **USERID** | **TIME** | **OPERATION** | **HISTORY** |
| ANONYMOUS | 12/22/2018 | Insert | Old materials: and Newmaterials: Aurum |

3. Триггер с When

Отражает изменение данных в таблице по условию When.

Создадим таблицу опытных инженеров, в которую будем записывать имя сотрудника, год рождения и его место работы.

create table Old\_engeneers (name varchar2(20), age int, firms varchar2(50));

Триггер Engineers\_age\_trigger будет срабатывать кажддый раз при новой записи в таблицу инженеров или обновлении года рождения инженера.

create or replace trigger Engineers\_age\_trigger

after insert or update on Engineers

for each row

when(new.YearOB<=1980)

begin

insert into Old\_engeneers

values (:new.EngName,:new.YearOB,:new.Name\_f);

End Engineers\_age\_trigger;

select \* from Engineers where Name\_f='Cala';

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PASS** | **ENGNAME** | **YEAROB** | **STARTOC** | **NAME\_F** |
| 227322 | Gabe | 1980 | 09/01/2010 | Cala |
| 673455 | George | 1975 | 09/01/2000 | Cala |

Для иллюстрации работы триггера добавим в таблицу Engineers нового инженера.

insert into Engineers values ( 337342, 'Martin Luter', 'IT1', 1980, to\_date('01.09.2010','dd.mm.yyyy'), 'Cala');

select \* from Old\_engeneers;

После совершения операции insert его данные были зафиксированы в таблице Old\_engeneers.

|  |  |  |
| --- | --- | --- |
| **ENGNAME** | **YEAROB** | **NAME\_F** |
| Martin Luter | 1980 | Cala |

4. Программирование клиента базы данных

Наиболее востребованным и распространенным способом организации работы баз данных является их программирование с помощью прикладных языков программирования. Одним из самых удобных способов управления данными является соединения SQL и JAVA на основе средств JDBC.

4.1 Первая транзакция

Позволяет в случае сбоя в системе, ошибки, возникновения исключения сохранить корректность вводимых данных за счет атомарности транзакции. Т.е. данные могут быть или внесены в БД полностью либо произойдет откат к предыдущему состоянию (до момента начала выполнения транзакции).

В конкретном примере транзакция транзакции производит удаление данных, связанных с конкретным инженером в ходе его увольнения. Происходит последовательное удаление из таблиц взаимосвязанных записей. В случае ошибки или несвоевременного удаления из какой-либо таблицы транзакция будет полностью отменена. Таким образом обеспечивается целостность данных.

public static void leaving (Connection con)throws SQLException {

String name="Mihail", SensName="";

int doc=0;

try {

con.setAutoCommit(false);

Statement st = con.createStatement();

ResultSet res=st.executeQuery("select Pass from Engineers where EngName='"+name+"'");

if(res.next())

doc = res.getInt(1);

res=st.executeQuery("select SensName from Sensors where Pass="+doc+"");

while(res.next()) {

SensName = res.getString(1);

}

st.executeUpdate("delete from Stands where SensName='"+SensName+"'");

st.executeUpdate("delete from Sensors where Pass="+doc+"");

st.executeUpdate("delete from Engineers where Pass="+doc+"");

con.commit();

} catch (SQLException e) {

e.printStackTrace();

con.rollback();

con.close();

}

}

4.2 Вторая транзакция

Для обеспечения параллельности одновременно выполняемых транзакции был выбран режим изоляции READ COMMITED, который устанавливается с помощью функции setTransactionIsolation() и позволяет запретить случаи потерянного обновления и грязного чтения данных. Этот уровень не защищает от ситуации, когда транзакция получает разные ответы на одни и те же запросы, но в этом режиме возможно обнаружение фантомной вставки.

В приведенном коде производится динамическое ценообразование стоимости складских услуг в зависимости от числа и хранящихся там материалов. В данном примере выбран именно этот уровень изоляции, поскольку он позволяет избежать эффекта чтения грязных, который возникнет при выполнении параллельно этой транзакции много раз над одним и тем же кортежем.

public static void value\_increase (Connection con) throws SQLException {

int Mprice=0;

int cost= 46, WarehID=0, count=0;

try {

con.setAutoCommit(false);

con.setTransactionIsolation(2);

Statement st = con.createStatement();

ResultSet res=st.executeQuery("select WarehID from Warehouses where Costs="+cost+"");

if(res.next()) {

WarehID = res.getInt(1);

}

res=st.executeQuery("select Mprice from Materials where WarehID="+WarehID+"");

while(res.next()) {

count++;

}

if(count>=2) {

cost=cost+5\*count;

st.executeUpdate("update Warehouses set Costs="+cost+" where WarehID='"+WarehID+"'");

for(int i=0; i<count; i++) {

Mprice+=30\*count;

st.executeUpdate("update Materials set Mprice="+Mprice+" where WarehID='"+WarehID+"'");

}

con.commit();

} else con.rollback();

} catch (SQLException e) {

e.printStackTrace();

con.rollback();

}

}

}

Заключение

За время написания курсовой работы была изучены специальные источники по затрагиваемой тематике, также были приобретены ключевые навыки работы с СУБД Oracle 11 g Express Edition, проектирования моделей баз данных с помощью моделей «сущность-связь» и . языка определения данных ODL, был получен необходимый опыт создание таблиц на языке SQL и написание к ним запросов. Также попутно были изучены основные приемы программирования базы данных с помощью языка PL\SQL, такие как курсоры и треггеры, и методов JDBS.

В первой части проделанной работы была создана ER-модель, реализующая визуально-графическое представление отношений и связей между ними, а также ODL модель базы данных датчиков измерения механических величин, представляющая все типы возможных связей отношений. Также было определенно, что база данных находится в нормальной форме Бойса-Кодда.

Во второй части был написан код загрузки базы данных на языке SQL, в отношения были введены первичные и внешние ключи, а также все необходимые типы SELECT-FROM-WHERE запросов с использованием различных соединений таблиц и пользовательских представлений.

В третьей части был реализован код загрузки триггеров 3 типов: триггер поддержания ссылочной целостности, триггер с ветвлением и триггер с условием when.

В последней части курсовой работы были разработаны две транзакции на языке программирования Java. В ней были написаны два основных типа транзакций: транзакция с атомарностью и транзакция с изоляцией. Для каждой транзакции был описан бизнес-процесс и обоснован метод обеспечения атомарности или изоляции.

Список источников

1. Малыхина Г.Ф. Управление данными: учеб. пособие. – СПб.: Изд-во Политехн. ун-та, 2012.252 с.
2. Нокс Д. Oracle Database 10g. Создание эффективной системы безопасности. – М.:Лори, 2007.-556с.
3. Лафоре Р. Структуры данных и алгоритмы в Java. Классика Computers Science. 2-е изд. - СПб.: Питер, 2013. — 704 с.:
4. <https://docs.oracle.com>
5. <https://stackoverflow.com/questions>
6. <https://habr.com>
7. <http://www.sql.ru/forum/>