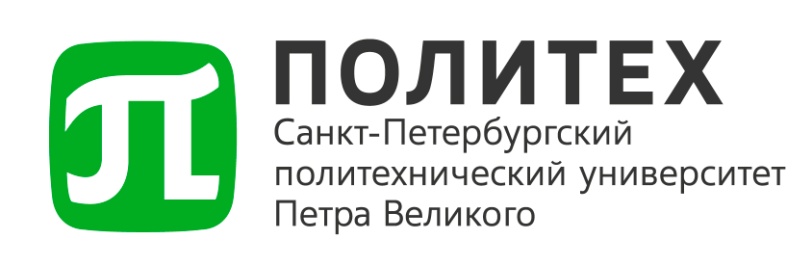
ФЕД­­ЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

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



**Курсовая работа**

**«База данных охранных датчиков»**

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

Выполнил

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

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

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Санкт-Петербург

2019

Оглавление

Введение 1

1. Моделирование данных 2

1.1 E/R-модель данных2

1.2 ODL-модель 4

2. База данных Oracle 7

2.1 Создание базы данных

охранных датчиков7

2.2 Запросы к базе данных

охранных датчиков 8

3. Триггеры базы данных 18

3.1 Триггер уровня строки, с псевдозаписями 18

3.2 Триггер с предикатами if-then19

3.3 Триггер с условием when 21

4. Программирование клиента базы данных22

4.1 Первая транзакция22

4.2 Вторая транзакция24

Заключение 25

Список источников 26

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.

firm

security sensors

cabinet

computer

employee

stored

working

sells

manages

buys

buyer

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 buyer (key: (passport data)) {

Attribute string name;

Attribute int passport data;

Attribute struct date {

Int day;

Int month;

Int year;

} d.o.b;

}

Relationship firm obey;

Inverse firm :: buys;

Interface firm (key:(account number)) {

Attribute int capital;

Attribute string director;

Attribute string name;

Attribute struct add {

string city;

string street;

int house;

} address;

Attribute int account number;

}

Relationship buyer buys;

Inverse buyer :: obey;

Relationship set< employee >working;

Inverse employee:: obey;

Relationship security sensors sells;

Inverse security sensors:: own;

Interface employee (key:(name)) {

Attribute struct add {

string city;

string street;

int house;

} address;

Attribute string name;

Attribute int salary;

Attribute struct date {

int day;

int month;

int year;

} d.o.b;

Attribute string specialty;

}

Relationship firm obey;

Inverse firm:: working;

Relationship set< computer > manages;

Inverse computer:: obey;

Interface computer (key:(serial number)) {

Attribute int cost;

Attribute int serial number;

Attribute string characteristics;

}

Relationship cabinet stored;

Inverse cabinet:: own;

Relationship set< employee >obey;

Inverse employee:: manages;

Interface cabinet (key:(number)) {

Attribute int space;

Attribute int number;

Attribute int number of computers;

}

Relationship set< computer > own;

Inverse computer:: stored;

Interface security sensors (key:( principle of operation)) {

Attribute string principle of operation;

Attribute string sensor name;

Attribute int cost;

Attribute int action range;

}

Relationship firm own;

Inverse firm:: sells;

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.

CREATE TABLE buyer (name varchar2(100), profession varchar2(50),

passport\_data int PRIMARY KEY, d\_o\_b DATE);

CREATE TABLE firm (capital int, director varchar2(100), name varchar2(100) ,

adress varchar2(100), account\_number int PRIMARY KEY);

CREATE TABLE cabinet (space int, number\_cab int PRIMARY KEY,

number\_of\_computers int);

CREATE TABLE security\_sensors (Sensor\_name varchar2(50), principle\_of\_operation varchar2(50) PRIMARY KEY,

cost int, action\_range int);

CREATE TABLE employee (name varchar2(100) PRIMARY KEY, adress varchar2(100),

salary int, d\_o\_b DATE, specialty varchar2(100),

account\_number int REFERENCES firm(account\_number));

CREATE TABLE computer (cost int, serial\_number int PRIMARY KEY, characteristics varchar2(50),

number\_cab int REFERENCES cabinet(number\_cab), name varchar2(100) REFERENCES employee(name));

CREATE TABLE Buys (passport\_data int REFERENCES buyer(passport\_data), account\_number int REFERENCES firm(account\_number));

CREATE TABLE Sells (principle\_of\_operation varchar2(50) REFERENCES security\_sensors(principle\_of\_operation),

account\_number int REFERENCES firm(account\_number));

2.2 Запросы к базе данных датчиков измерения механических величин

1. Простой запрос SELECT-FROM-WHERE к одной таблице – 1 запрос.

Вывести все характеристики компьютеров, стоящих больше 35000.

select characteristics from computer where cost > 35000;

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |
| --- |
| **CHARACTERISTICS** |
| Intel core i7, Nvidia GeForse 1050 |
| Intel core i7, Nvidia GeForse 2080 |
| Intel core i5, Nvidia GeForse 1080 |

2. Простой запрос SELECT-FROM-WHERE к двум таблицам – 1 запрос.

Вывести все фирмы, которым принадлежит детектор дыма

select name from firm, Sells where firm.account\_number=Sells.account\_number and principle\_of\_operation='Smoke detector'

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |
| --- |
| **NAME** |
| Abright |

3. Простой запрос SELECT-FROM-WHERE к трем таблицам – 1 запрос

|  |  |  |  |
| --- | --- | --- | --- |
| Вывести имя сотрудника и его рабочий компьютер, для компании 'PolyTech' |  |  |  |
| select employee.name, serial\_number from computer, employee, firm where computer.name=employee.name and employee.account\_number=firm.account\_number and firm.name='PolyTech'   |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  |  |  |  | | --- | --- | | **NAME** | **SERIAL\_NUMBER** | | Merkushev Leonard | 1915003 | | Shcherbakov Cornelius | 1915004 | | Kuzmin Donat | 1915005 | |  |  |  |
|  |  |  |  |

4. Запрос к одной таблице, умноженной на себя – 1 запрос

Вывести сенсоры и диапазон действия для тех устройств, у которых он одинаков.

Select DISTINCT A1.Sensor\_name, A2.Sensor\_name, A1.action\_range FROM security\_sensors A1, security\_sensors A2 WHERE A1.action\_range = A2.action\_range and A1.Sensor\_name!= A2.Sensor\_name;

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **SENSOR\_NAME** | **SENSOR\_NAME** | **ACTION\_RANGE** |
| Contact | Vibrating | 1 |
| Vibrating | Contact | 1 |
| Acoustic | Other | 10 |
| Other | Acoustic | 10 |

5. Запросы с подзапросами.

Использующие 2 таблицы - 2 запроса

Вывести специальности и их зарплаты в компании PolyTech.

|  |  |  |  |
| --- | --- | --- | --- |
| Select specialty, salary from employee where employee.account\_number=(select firm.account\_number from firm where firm.name='PolyTech') |  |  |  |

|  |  |
| --- | --- |
| **SPECIALTY** | **SALARY** |
| Tester | 60000 |
| Accountant | 45000 |
| Copywriter | 39000 |

Вывести имя и адрес сотрудника, работающего под руководством Nikolay Beresnev.

Select employee.name, employee.adress from employee where employee.account\_number=(Select firm.account\_number from firm where director='Nikolay Beresnev')

|  |  |
| --- | --- |
| **NAME** | **ADRESS** |
| Lebedev Bronislav | Moscow, Meadow Street, 32 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Использующие 3 таблицы - 2 запроса.

Вывести имя покупателя и его дату рождения для фирмы 'Abright'

|  |  |  |  |
| --- | --- | --- | --- |
| select buyer.name, d\_o\_b from buyer where passport\_data=(select Buys.passport\_data from Buys where Buys.account\_number=(select firm.account\_number from firm where firm.name=' Abright')) |  |  |  |

Вывести имена и профессии покупателей 'Radio wave.

|  |  |  |  |
| --- | --- | --- | --- |
| Select buyer.name, profession from buyer where buyer.passport\_data=(select Buys.passport\_data from Buys where Buys.account\_number=(select Sells.account\_number from Sells where principle\_of\_operation ='Radio wave')) |  |  |  |

|  |  |
| --- | --- |
| **NAME** | **PROFESSION** |
| Vlad | travel agent |

Каждый из запросов выполняется дважды используя слово из (EXISTS, IN) и/или из (ALL,

ANY). Все слова должны быть в запросах использованы – 4 запроса. Каждый запрос сделайте

дважды, используя (EXISTS, IN), используя (ALL, ANY).

Запросы с использование пары (EXISTS, IN)

Вывести регистрационные номера фирм, у которых покупатели бизнесмены.

|  |  |  |  |
| --- | --- | --- | --- |
| Select account\_number from Buys where Buys.passport\_data IN (Select passport\_data from buyer where profession='businessman') |  |  |  |

|  |
| --- |
| **ACCOUNT\_NUMBER** |
| 1412001 |
| 1412011 |
| 1412045 |

Select account\_number from Buys where exists (Select \* from buyer where profession='businessman' and Buys.passport\_data=buyer. passport\_data)

|  |
| --- |
| **ACCOUNT\_NUMBER** |
| 1412045 |
| 1412011 |
| 1412001 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Запросы с использование пары (ALL, ANY)

Найти названия, стоимость и принцип работы сенсоров, которые пока то не принадлежат ни одной фирме.

|  |  |  |  |
| --- | --- | --- | --- |
| Select Sensor\_name, cost, security\_sensors.principle\_of\_operation from security\_sensors where security\_sensors.principle\_of\_operation <>ALL(select Sells.principle\_of\_operation from Sells) |  |  |  |

|  |  |  |
| --- | --- | --- |
| **SENSOR\_NAME** | **COST** | **PRINCIPLE\_OF\_OPERATION** |
| Movements | 17000 | Infrared |
| Movements | 8000 | Ultrasonic |

Select Sensor\_name, cost, security\_sensors.principle\_of\_operation from security\_sensors where not security\_sensors.principle\_of\_operation =ANY(select Sells.principle\_of\_operation from Sells)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **SENSOR\_NAME** | **COST** | **PRINCIPLE\_OF\_OPERATION** |
| Movements | 17000 | Infrared |
| Movements | 8000 | Ultrasonic |

6. Запросы теории множеств UNION, INTERSECT, EXCEPT(MINUS) – 3 запроса

Запрос с использованием INTERSECT

Вывести номера аккаунтов, у которых есть и покупатели и сотрудники.

select account\_number from Buys INTERSECT select account\_number from Sells

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |
| --- |
| **ACCOUNT\_NUMBER** |
| 1412001 |
| 1412011 |
| 1412045 |
| 1412068 |
| 1412099 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Запрос с использованием EXCEPT(MINUS)

Вывести названия сенсоров, которые не реализуются фирмами.

select principle\_of\_operation from security\_sensors minus select principle\_of\_operation from Sells

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |
| --- |
| **PRINCIPLE\_OF\_OPERATION** |
| Infrared |
| Ultrasonic |

Запрос с использованием UNION

Вывести все профессии, задействованные в базе данных

select profession from buyer union select specialty profession from employee

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **PROFESSION** | | | |
| Accountant | | | |
| Call-center operator | | | |
| Chief Engineer | | | |
| Cleaner | | | |
| Copywriter | | | |
| PR Manager | | | |
| Sales Representative | | | |
| Secretary-Referen | | | |
| Tester | | | |
| Web-programmer | | | |
| air controller | | | |
| auto mechanic | | | |
| Businessman | | | |
| Contractor | | | |
| Developer | | | |
| Doctor | | | |
| Freelancer | | | |
| Householder | | | |
| Lawyer | | | |
| travel agent | | | |
|  |  |  |  |

7. Пользовательское представление из 3 таблиц. 1 запрос к представлению – 1 запрос.

Создать пользовательское представление, где связаны данные логина пользователя,

принадлежащее ему место в хранилище и компания, сотрудником которой он является.

CREATE VIEW Buyer\_Sensor AS (Select buyer.name Buyer, security\_sensors.principle\_of\_operation Sensor, cost

FROM security\_sensors, buyer, Buys, Sells

WHERE Sells.account\_number=Buys.account\_number and security\_sensors.principle\_of\_operation=Sells.principle\_of\_operation and buyer.passport\_data=Buys.passport\_data)

Вывести всю информацию о клиентах, сенсоры которых стоят больше 1000

Select \* from Buyer\_Sensor where cost >1000

|  |  |  |
| --- | --- | --- |
| **BUYER** | **SENSOR** | **COST** |
| Kate | Opening sensor | 4000 |
| Kate | Opening sensor | 4000 |
| Vlad | Radio wave | 15000 |
| Vlad | Smoke detector | 4000 |

8. Внешние и внутренние соединения (JOIN ON, CROSS JOIN, NATURAL OUTER

JOIN,OUTER , NATURAL LEFT OUTER, NATURAL RIGHT OUTER ) – 6 запросов

Запрос с использованием JOIN ON.

Вывести названия фирм, которым принадлежит Tilt sensor

select name from firm JOIN Sells ON firm.account\_number=Sells.account\_number where principle\_of\_operation= 'Tilt sensor'

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |
| --- |
| **NAME** |
| PolyTech |
| TD "Leader-SB" |

Запрос с использованием CROSS JOIN.

Вывести имена и профессии сотрудников, зарплаты которых не хватит на покупку их рабочего компьютера.

select employee.name,specialty, salary His\_sal, cost Cost\_of\_comp from employee CROSS JOIN computer where employee.name=computer.name and salary<cost

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **NAME** | | | | **SPECIALTY** | **HIS\_SAL** | **COST\_OF\_COMP** |
| Lebedev Bronislav | | | | Cleaner | 32000 | 39000 |
|  |  |  |  |

Запрос с использованием NATURAL JOIN.

Вывести имена и номера покупателей, являющихся клиентами и фирм и работающих турагентами.

Select name, account\_number from Buys NATURAL JOIN buyer where profession='travel agent'

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **NAME** | **ACCOUNT\_NUMBER** | | Vlad | 1412031 | | Vlad | 1412099 |  |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |

Запрос с использованием FULL OUTER JOIN.

Вывести паспорта покупателей и приобретенные ими сенсоры.

Select passport\_data, principle\_of\_operation from Buys FULL JOIN Sells ON Sells.account\_number=Buys.account\_number

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PASSPORT\_DATA** | | | | **PRINCIPLE\_OF\_OPERATION** |
| 651190 | | | | Opening sensor |
| 651190 | | | | Opening sensor |
| 651190 | | | | Tilt sensor |
| 851236 | | | | Tilt sensor |
| 369888 | | | | Tilt sensor |
| 121666 | | | | Tilt sensor |
| - | | | | Glass break sensor |
| - | | | | Capacitive sensor |
| - | | | | Beam sensor |
| 260308 | | | | Radio wave |
| 260308 | | | | Smoke detector |
| 260308 | | | | Microphone |
| 260308 | | | | - |
| 158168 | | | | - |
| 260862 | | | | - |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Запрос с использованием RIGHT OUTER JOIN.

Вывести номера аккаунтов фирм и названия распространяемых ими сенсоров.

Select DISTINCT Sells.account\_number,Sensor\_name from Sells RIGHT JOIN security\_sensors ON security\_sensors.principle\_of\_operation=Sells.principle\_of\_operation

|  |  |
| --- | --- |
| **ACCOUNT\_NUMBER** | **SENSOR\_NAME** |
| 1412044 | Other |
| 1412099 | Other |
| 1412011 | Contact |
| - | Movements |
| 1412044 | Vibrating |
| 1412099 | Movements |
| 1412099 | Acoustic |
| 1412045 | Vibrating |
| 1412068 | Vibrating |

Запрос с использованием LEFT OUTER JOIN.

Вывести номера кабинетов, их площадь и серийные номера стоящих там компьютеров, для кабинетов больших 70 кв.м.

Select cabinet.number\_cab, space, serial\_number from cabinet LEFT JOIN computer ON computer.number\_cab=cabinet.number\_cab where space>70

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| **NUMBER\_CAB** | **SPACE** | **SERIAL\_NUMBER** |
| 5 | 73 | 1915005 |
| 5 | 73 | 1915006 |
| 5 | 73 | 1915007 |
| 7 | 135 | 1915008 |
| 7 | 135 | 1915009 |
| 4 | 145 | - |
| 3 | 254 | - |
| 9 | 87 | - |

9. Агрегация (SUM,COUNT,AVG,MIN,MAX) – 1 запрос.

Запрос с использованием AVG.

Вывести имя и зарплату наиболее высокооплачиваемого сотрудника.

|  |  |  |  |
| --- | --- | --- | --- |
| SELECT name, salary FROM employee WHERE salary=(SELECT MAX(salary) FROM employee) |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| **NAME** | **SALARY** |
| Bobrov Klim | 75600 |

10. Агрегация и GROUP BY - 1 запрос.

Вывести все открытые компании и число их сотрудников.

Select firm.name, COUNT(employee.name) FROM firm,employee where employee.account\_number=firm.account\_number GROUP BY firm.name;

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |
| --- | --- |
| **NAME** | **COUNT(EMPLOYEE.NAME)** |
| ARMO-Systems | 2 |
| iQ Comfort | 2 |
| PolyTech | 3 |
| Abright | 2 |
| Alliance-Inter | 1 |

11. Агрегация, GROUP BY, HAVING - 1 запрос.

Подсчитать число специалистов, возраст которых больше сорока.

SELECT specialty, Count(specialty) FROM employee GROUP BY specialty HAVING MIN(d\_o\_b)<'01-01-1979'

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

|  |  |
| --- | --- |
| **SPECIALTY** | **COUNT(SPECIALTY)** |
| Chief Engineer | 1 |
| Web-programmer | 1 |
| Secretary-Referen | 1 |

12. Коррелированный запрос – 2 запроса разного типа.

Найти сотрудников, работающих в фирме Армо-Системс.

SELECT employee.name FROM employee WHERE EXISTS( SELECT \* FROM firm WHERE firm.account\_number=employee.account\_number AND firm.name='ARMO-Systems')

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | |
| |  | | --- | | **NAME** | | Kulakov Kondrat | | Solovyov Julius |  |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | | | | | | | | | | | | | | |
|  |  |  |  |  | |  |  |  |  |  |  |  |

Найти серийные номера компьютеров, находящихся в одном кабинете.

|  |  |  |  |
| --- | --- | --- | --- |
| Select DISTINCT A1.serial\_number, A1.number\_cab FROM computer A1 WHERE number\_cab IN (Select A2.number\_cab FROM computer A2 WHERE A1.serial\_number<>A2.serial\_number) |  |  |  |

|  |  |
| --- | --- |
| **SERIAL\_NUMBER** | **NUMBER\_CAB** |
| 1915002 | 1 |
| 1915001 | 1 |
| 1915004 | 2 |
| 1915003 | 2 |
| 1915007 | 5 |
| 1915006 | 5 |
| 1915005 | 5 |
| 1915009 | 7 |
| 1915008 | 7 |

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 employee триггер обновляет связанные с этой таблицей дочерние записи в таблице computer, на которые есть ссылка.

create or replace trigger Employee\_trigger

after update of name on employee

for each row

begin

update computer set name=:new.name where name=:old.name;

end Employee\_trigger;

Приведём пример срабатывания триггера при обновлении данных в таблице employee. В данном случае данные также обновляются и в дочерних таблицах.

select \* from computer where name='Solovyov Julius'

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **COST** | **SERIAL\_NUMBER** | **CHARACTERISTICS** | **NUMBER\_CAB** | **NAME** |
| 35000 | 1915001 | Intel core i5, Nvidia GeForse 970 | 1 | Solovyov Julius |

update employee set name='sucpect' where name='Solovyov Julius'

select \* from computer where name='sucpect'

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **COST** | **SERIAL\_NUMBER** | **CHARACTERISTICS** | **NUMBER\_CAB** | **NAME** |
| 35000 | 1915001 | Intel core i5, Nvidia GeForse 970 | 1 | sucpect |

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

Создадим новую таблицу аудита компьютеров, 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 Computer\_audit(userID varchar2 (30), time Date, operation varchar2 (20), history varchar2(60));

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

create or replace trigger Computer\_trigger

after insert or delete or update on computer

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 Computer\_audit values (user, sysdate, operation ,

'Old serial number: '||:old.serial\_number||' New serial nubmer: '||:new.serial\_number);

end Computer\_trigger;

В качестве примера удалим все компьютеры дороже 34000.

delete from computer where cost>34000

Выведем все записи таблицы Computer\_audit.

select \* from Computer\_audit

|  |  |  |  |
| --- | --- | --- | --- |
| **USERID** | **TIME** | **OPERATION** | **HISTORY** |
| ANONYMOUS | 01/31/2019 | Delete | Old serial number: 1915001 New serial nubmer: |
| ANONYMOUS | 01/31/2019 | Delete | Old serial number: 1915003 New serial nubmer: |
| ANONYMOUS | 01/31/2019 | Delete | Old serial number: 1915009 New serial nubmer: |
| ANONYMOUS | 01/31/2019 | Delete | Old serial number: 1915010 New serial nubmer: |

3. Триггер с When

Отражает изменение данных в таблице по условию When.

Создадим таблицу молодых сотрудников, в которую будем записывать фио, дату рождения и размер зарплаты.

create table Young\_employer (name varchar2(20), d\_o\_b DATE, salary int);

Триггер Employee\_age\_trigger будет срабатывать каждый раз при новой записи в таблицу инженеров или обновлении года рождения специалиста.

create or replace trigger Employee\_age\_trigger

after insert or update on employee

for each row

when(new.d\_o\_b>'01-01-1979')

begin

insert into Young\_employer

values (:new.name,:new.d\_o\_b,:new.salary);

End Employee\_age\_trigger;

Выведем все данные сотрудников фирмы с номером 1412001

select \* from employee where account\_number=1412001

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NAME** | **ADRESS** | **SALARY** | **D\_O\_B** | **SPECIALTY** | **ACCOUNT\_NUMBER** |
| Sucpect | Moscow, Sovetskaya Street, 84 | 65000 | 10/12/1989 | Call-center operator | 1412001 |
| Kulakov Kondrat | Moscow, Central Street, 80 | 50000 | 02/10/1970 | Secretary-Referen | 1412001 |

Для иллюстрации работы триггера добавим в таблицу employee нового сотрудника, эта вставка будет зафиксирована в таблице Young\_employer.

INSERT INTO employee VALUES ('Harper Bob', 'Moscow, Red line Street, 30', 60000, TO\_DATE('11-04-1984','dd-mm-yyyy'), 'Anykeyer', 1412001);

select \* from Young\_employer

|  |  |  |
| --- | --- | --- |
| **NAME** | **D\_O\_B** | **SALARY** |
| Harper Bob | 04/11/1984 | 60000 |

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 перевод денег со счета одного человека другому. Происходит последовательное считывание данных из записях об этих сотрудниках, получение и измерение баланса на их счету. В случае ошибки транзакция будет полностью отменена. Таким образом обеспечивается целостность данных.

public static void zaym(Connection con) throws SQLException {

String name1="Merkushev Leonard", name2="Kulakov Kondrat

int bal1=0, bal2=0;

try {

con.setAutoCommit(false);

Statement st=con.createStatement();

String SQL = "select salary from employee where name='"+name1+"'";

ResultSet dolzhnik=st.executeQuery(SQL);

if (dolzhnik.next())

bal1=dolzhnik.getInt(1);

SQL = "select salary from employee where name='"+name2+"'";

ResultSet zaemschik=st.executeQuery(SQL);

if (zaemschik.next())

bal2=zaemschik.getInt(1);

System.out.println(name1+" \t"+bal1);

System.out.println(name2+" \t"+bal2);

System.out.println("Enter Transaction Amount");

Scanner s = new Scanner(System.in);

int money = s.nextInt();

if(money<bal2) {

bal2=bal2-money;

st.executeUpdate("update employee set salary ="+bal2+" where name='"+name2+"'");

bal1=bal1+money;

st.executeUpdate("update employee set salary="+bal1+" where name='"+name1+"'");

con.commit();

}

dolzhnik=st.executeQuery("select salary from employee where name='"+name1+"'");

if (dolzhnik.next())

bal1=dolzhnik.getInt(1);

zaemschik=st.executeQuery("select salary from employee where name='"+name2+"'");

if (zaemschik.next())

bal2=zaemschik.getInt(1);

System.out.println(name1+" \t"+bal1);

System.out.println(name2+" \t"+bal2);

} 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 удаление всех записей и информации о определенной фирме. 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 delete\_firm (Connection con) throws SQLException {

String firma=" Abright";

ArrayList <String> name = new <String> ArrayList();

int number=0, cab=0;

try {

con.setAutoCommit(false);

con.setTransactionIsolation(2);

Statement st=con.createStatement();

ResultSet res=st.executeQuery("select account\_number from firm where name='"+firma+"'");

if(res.next())

number=res.getInt(1);

System.out.println("Company "+firma+"\t "+number);

st.executeUpdate("delete from Sells where account\_number="+number);

st.executeUpdate("delete from Buys where account\_number="+number);

res=st.executeQuery("select name from employee where account\_number="+number);

while(res.next())

name.add(res.getString(1));

for(int i=0; i<name.size(); i++)

st.executeUpdate("delete from computer where name='"+name.get(i)+"'");

st.executeUpdate("delete from employee where account\_number="+number);

st.executeUpdate("delete from firm where account\_number="+number);

con.commit();

res=st.executeQuery("select account\_number from firm where name='"+firma+"'");

if(res.next()) {

number=res.getInt(1);

System.out.println("Company "+firma+"\t "+number);

}

else System.out.println("Empty"); } 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/>