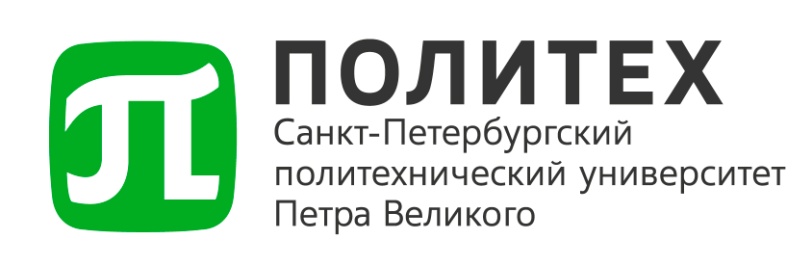
ФЕД­­ЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

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



**Курсовая работа**

**“База данных облачного хранилища”**

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

Выполнил

студент гр.23535/2 Летюшев А. И.

Преподаватель Малыхина Г. Ф.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2018 г.

Санкт-Петербург

2018

Оглавление

Введение 1

1. Моделирование данных 2

1.1 E/R-модель данных3

1.2 ODL-модель 4

2. База данных Oracle 7

2.1 Создание базы данных облачных хранилищ 7

2.2 Запросы к базе данных облачных хранилищ 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.

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.

1.1 ER-модель данных

Cloud

use

Cluster

Server

rule

work

Resource center

save

regulate

belong

own

Calculation

Data storage

Account

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 Cloud (key name)

{

Attribute string name;

Attribute string tech;

Attribute string type;

Attribute string company;

}

relationship resource center use;

inverse resource center::obey;

Interface Resource center (key city)

{

Attribute string director;

Attribute string city;

Attribute int numb nodes;

}

relationship cloud obey;

inverse cloud::use;

relationship set <cluster> rule;

inverse cluster::conform;

relationship set <data storage> save;

inverse data storage::control;

relationship set<calculation> regulate;

inverse calculation::perform;

Interface Cluster (key ID, address)

{ Attribute int ID;

Attribute struct add {

string street;

int house;

}address;

Attribute int life time;

Attribute int numb servers;

}

relationship resource center conform;

inverse resource center::rule;

relationship set <server> work;

inverse server::keep;

Interface Server (key serial\_numb)

{

Attribute int serial\_numb;

Attribute float mem\_size;

Attribute float power;

}

relationship cluster keep;

inverse cluster::work;

Interface Data storage (key cost, space)

{

Attribute float cost;

Attribute float space;

Attribute float available;

}

relationship resource center control;

inverse resource center::save;

relationship account belong;

inverse account::make;

Interface Сalculation (key price)

{

Attribute float price;

Attribute float speed;

Attribute float consumption;

Attribute string client;

}

relationship resource center perform;

inverse resource center::regulate;

relationship set<account> practice;

inverse account::own;

Interface Account (key login, password)

{

Attribute string login;

Attribute string password;

Attribute int subs;

Attribute struct data {

int year; month, day;

} d.o.b.;

}

relationship data storage make;

inverse data storage belong;

relationship set <calculation> own;

inverse calculation::practice;

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

Oracle Database – одна из лучших СУБД, кроссплатформенная и кибербезопасная, эта объектно-реляционная система заслуженно является самой популярной и востребованной в мире.

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 облачных хранилищ и инициализация таблиц методом insert

Оператор CREATE TABLE позволяет создавать новые таблицы данных, которые могут быть заполнены данными с помощью оператора INSERT или удалены с помощью DROP. SQL-запросы основаны на предложении SELECT-FROM-WHERE, применяемое к одному или объединению отношений

CREATE TABLE Cloud (name varchar2(100) PRIMARY KEY, tech varchar2(50), type\_с varchar2(100),

company varchar2(200));

CREATE TABLE Resource\_center (director varchar2(100), city varchar2(30) PRIMARY KEY,

numb\_nodes int, name varchar2(100) REFERENCES Cloud (name));

CREATE TABLE Cluster\_c (ID varchar2(20) PRIMARY KEY, city varchar2(30) REFERENCES Resource\_center (city),

address varchar2(100), numb\_servers int, life\_time int);

CREATE TABLE Server (serial\_numb int PRIMARY KEY, mem\_size float, power\_c int,

ID\_C varchar2(20) REFERENCES Cluster\_c(ID));

CREATE TABLE Data\_storage (cost int, space\_c int PRIMARY KEY, available float,

city varchar2(30) REFERENCES Resource\_center (city));

CREATE TABLE Calculation (client varchar2(50) PRIMARY KEY,

city varchar2(30) REFERENCES Resource\_center (city), price float,

speed float, cosumption int);

CREATE TABLE Account (login varchar2(50), password varchar2(20), subs varchar2(16),

d\_o\_b DATE, client varchar2(50) REFERENCES Calculation (client),

space\_c int REFERENCES Data\_storage (space\_c),PRIMARY KEY(login, password) );

2.2 Запросы к базе данных облачных хранилищ

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

Вывести все компании у которых корпоративные хранилища.

SELECT company FROM Cloud Corporate\_Cloud WHERE type\_с = 'corporate';

Πcompany(ϭ type\_с = 'corporate' ( Cloud))

|  |
| --- |
| Microsoft |

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

Вывести названия хранилищ, использующих технологию SaaS и число узлов больше 15.

SELECT Cloud.name FROM Cloud, Resource\_center WHERE tech = 'SaaS' and numb\_nodes > 15 and Resource\_center.name = Cloud.name;

ΠCloud.name (ϭ tech = 'SaaS' ( Cloud) >< ϭ numb\_nodes > 15 (Resource\_center))

|  |
| --- |
| Google Drive |

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

Вывести наименования всех аккаунтов, клиентов и принадлежащее им место, где подключено более чем 500 ГБ

Select Account.login, Calculation.client, Account.space\_c FROM Account, Calculation, Data\_storage

WHERE Account.client = Calculation.client and Account.space\_c = Data\_storage.space\_c and Account.space\_c>500;

Π Account.login, Calculation.client, Account.space\_c( Calculation >< ϭ Account.space\_c>500 ( Account)><Data\_storage)

|  |  |  |
| --- | --- | --- |
| Nine\_lit | JetBrains | 1020 |
| azimuth\_vr | SEMrush | 12000 |
| Gloria-Melman | SEMrush | 12000 |
| Martin-King | SEMrush | 12000 |

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

Вывести названия всех хранилищ, использующих одну и ту же технологию

Select DISTINCT A1.name, A2.name FROM Cloud A1, Cloud A2 WHERE A1.tech = A2.tech and A1.name != A2.name GROUP BY A1.name, A2.name;

**ρ**A1(Cloud)

**ρ**A2(Cloud)

Π A1.name, A2.name (A1 >< (A1.tech = A2.tech and A1.name != A2.name) A2)

|  |  |
| --- | --- |
| Azure | NextCloud |
| Box | Dropbox |
| Box | ICloud |
| Cloud Mail.Ru | Mega |
| Cloud Mail.Ru | OneDrive |
| Cloud Mail.Ru | Google Drive |
| Dropbox | ICloud |
| Dropbox | Box |
| Google Drive | Cloud Mail.Ru |
| Google Drive | OneDrive |
| Google Drive | Mega |
| ICloud | Dropbox |
| ICloud | Box |
| Mega | Cloud Mail.Ru |
| Mega | Google Drive |
| Mega | OneDrive |
| NextCloud | Azure |
| OneDrive | Mega |
| OneDrive | Google Drive |
| OneDrive | Cloud Mail.Ru |
| Azure | NextCloud |
| Box | Dropbox |
| Box | ICloud |
| Cloud Mail.Ru | Mega |
| Cloud Mail.Ru | OneDrive |
| Cloud Mail.Ru | Google Drive |
| Dropbox | ICloud |
| Dropbox | Box |
| Google Drive | Cloud Mail.Ru |
| Google Drive | OneDrive |
| Google Drive | Mega |
| ICloud | Dropbox |
| ICloud | Box |
| Mega | Cloud Mail.Ru |
| Mega | Google Drive |
| Mega | OneDrive |
| NextCloud | Azure |
| OneDrive | Mega |
| OneDrive | Google Drive |
| OneDrive | Cloud Mail.Ru |

5. Запросы с подзапросами.

Использующие 2 таблицы - 2 запроса

/\*Вывести ID кластеров, находящихся под управлением Яна Бехера\*/

Select Cluster\_c.ID FROM Cluster\_c

WHERE Cluster\_c.city = (Select Resource\_center.city

FROM Resource\_center WHERE director = 'Jan Becher');

Π Cluster\_c.ID (ϭ Cluster\_c.city = (Π Resource\_center.city (Resource\_center) (ϭ director = 'Jan Becher'(Resource\_center) ) ) (Cluster\_c)

|  |
| --- |
| 234.34.452 |
| 54.454.78 |
| 784.41.346 |

/\*Вывести названия всех фирм, производящих вычисления в OneDrive\*/

Select Calculation.client FROM Calculation

WHERE Calculation.city = (Select Resource\_center.city

FROM Resource\_center WHERE name = 'Azure');

Π Calculation.client (ϭ Calculation.city = (Π Resource\_center.city (Resource\_center) (ϭ name = 'Azure' (Resource\_center) ) ) (Calculation)

|  |
| --- |
| SEMrush |
| Epam |

Использующие 3 таблицы - 2 запроса.

Вывести серийные номера и мощность всех серверов, находящихся в праге на улице Вэ Смекач

SELECT serial\_numb FROM Server WHERE ID\_C =

(SELECT ID From Cluster\_c WHERE Cluster\_c.city =

(Select Resource\_center.city FROM Resource\_center

WHERE Resource\_center.city = 'Praga' and address = 'Ve Smeckach, 29'));

Π serial\_numb (Server) (ϭ ID\_C = (ΠID (ϭ Cluster\_c.city =(Π Resource\_center.city(ϭ Resource\_center.city = 'Praga' and address = 'Ve Smeckach, 29' (Resource\_center) ) ) (Cluster\_c) ) (Server)

|  |  |
| --- | --- |
| 84949498 | 15 |
| 94498498 | 22 |
| 49884998 | 15 |
| 68449844 | 22 |

Вывести названия всех компаний, сотрудничающих с Мегааплоад

Select client FROM Calculation

WHERE Calculation.city = (Select city From Resource\_center WHERE Resource\_center.name =

(Select Cloud.name FROM Cloud WHERE company = 'Megaupload'));

Π client (ϭ Calculation.city = (Π city (ϭ Resource\_center.name =(Π Cloud.name (Cloud) (ϭ company = 'Megaupload' (Resource\_center) ) ) (Resource\_center)) (Calculation)

|  |
| --- |
| Parallels |
| Wrike |
| Luxsoft |

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)

Найти директоров, под управлением которых находится больше 100 серверов

Select Resource\_center.director

FROM Resource\_center

WHERE Resource\_center.city IN (Select Cluster\_c.city

FROM Cluster\_c WHERE numb\_servers > 100);

|  |
| --- |
| Jan Becher |
| Daniil Lovki |
| Vlad Prykov |
| Hanz Lyahter |
| John Leon |

Select Resource\_center.director

FROM Resource\_center

WHERE EXISTS (Select \*

FROM Cluster\_c WHERE numb\_servers > 100 and Resource\_center.city = Cluster\_c.city);

|  |
| --- |
| Jan Becher |
| Daniil Lovki |
| Vlad Prykov |
| Hanz Lyahter |
| John Leon |

Запросы с использование пары (ALL, ANY)

Найти директоров, под ответственностью которых не лежат вычисления, потребляющие больше 5000 КВ\*ч

Select director FROM Resource\_center

WHERE Resource\_center.city <> ALL (Select Calculation.city

FROM Calculation WHERE cosumption > 5000);

|  |
| --- |
| Rahuul Amrit |
| John Leon |
| Vlad Prykov |
| Nathan Malak |
| David Richards |
| Daniil Lovki |
| Hanz Lyahter |

Select director FROM Resource\_center

WHERE NOT Resource\_center.city =ANY (Select Calculation.city

FROM Calculation WHERE cosumption > 5000);

|  |
| --- |
| Rahuul Amrit |
| John Leon |
| Vlad Prykov |
| Nathan Malak |
| David Richards |
| Daniil Lovki |
| Hanz Lyahter |

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

Запрос с использованием INTERSECT

Вывести города, в которых одновременно происходит хранение данных и выполнение вычислений

SELECT city FROM Data\_storage INTERSECT SELECT city FROM Calculation;

Π city (Data\_storage) **∩** Π city (Calculation)

|  |
| --- |
| Berlin |
| New-York |
| Praga |
| Saint-Petersburg |

Запрос с использованием EXCEPT(MINUS)

Вывести города, ресурсные центры которых не учавствуют в вычислениях

Select city FROM Resource\_center MINUS Select city FROM Calculation;

Π city (Recource\_center) **/** Π city (Calculation)

|  |
| --- |
| Amsterdam |
| Delhi |
| Kazan |
| London |
| Mosсow |
| San Francisco |

Запрос с использованием UNION

Вывести стоимость планных услуг всех облачных хранилищ в порядке убывания

Select cost FROM Data\_storage UNION Select price cost FROM Calculation ORDER BY cost DESC;

Π cost (Data\_storage) **U** Π **ρ**cost(price) (Calculation)

|  |
| --- |
| 1457.0999755859375 |
| 790.4000244140625 |
| 782.1400146484375 |
| 267.8999938964844 |
| 170.8000030517578 |
| 130.60000610351562 |
| 125.66000366210938 |
| 100 |
| 91.4000015258789 |
| 89.30000305175781 |
| 12 |
| 10 |
| 6 |
| 4 |
| 3 |
| 2 |
| 1 |
| 0 |

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 Account\_Data\_Calc AS (Select login, Calculation.client, Data\_storage.space\_c

FROM Account, Calculation, Data\_storage

WHERE Account.space\_c = Data\_storage.space\_c AND Account.client = Calculation.client);

Вывести все логины пользователей, являющихся сотрудниками фирмы Wrike

Select login FROM Account\_Data\_Calc WHERE client = 'Wrike';

|  |
| --- |
| Bravo\_22 |
| SPbSTU |
| bremya |

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 логины, дни рождения и города, в которых работают сотрудники фирм

Select login, d\_o\_b, city FROM Account JOIN Calculation ON Account.client = Calculation.client;

Π login, d\_o\_b, city( Account >< ϭ Account.client=Calculation.client (Calculation)

|  |  |  |
| --- | --- | --- |
| bremya | 11/24/1976 | Praga |
| SPbSTU | 07/02/1982 | Praga |
| Bravo\_22 | 06/15/1999 | Praga |
| Julia | 09/05/1999 | Berlin |
| berserk | 06/03/1995 | Berlin |
| Nine\_lit | 04/26/1991 | Saint-Petersburg |
| Martin-King | 01/08/2004 | New-York |
| Gloria-Melman | 02/07/2001 | New-York |
| azimuth\_vr | 08/09/1998 | New-York |
| rukiaa | 03/17/1993 | New-York |

Запрос с использованием CROSS JOIN.

Вывести всех директоров ресурсных центров с адресами и числом серверов подконтрольных им кластеров.

Select director, Resource\_center.city, address, numb\_servers FROM Resource\_center CROSS JOIN Cluster\_c

WHERE Resource\_center.city = Cluster\_c.city;

Π director, Resource\_center.city, address, numb\_servers ( ϭ Recource\_center.city=Cluster\_c.city(Recource\_center)>< ϭ Cluster\_c.city=Recource\_center.city(Cluster\_c))

|  |  |  |  |
| --- | --- | --- | --- |
| Jan Becher | Praga | Ve Smeckach, 29 | 64 |
| Jan Becher | Praga | Ujezd, 22 | 79 |
| Jan Becher | Praga | Yanski vreshek., 8 | 164 |
| Sergey Khrustin | Saint-Petersburg | Ligovckiy pr-t, 64 | 127 |
| Sergey Khrustin | Saint-Petersburg | Sadovaya, 123 | 63 |
| Hanz Lyahter | Berlin | Unter-den-Linden, 22a | 117 |
| Hanz Lyahter | Berlin | Hose str, 123 | 75 |
| David Richards | London | Baker str, 22b | 58 |
| John Leon | San Francisco | Staner str, 35 | 142 |
| Rick Sunches | New-York | Ager-roud, 184 | 81 |

Запрос с использованием NATURAL JOIN.

Вывести ID и полный адрес каждого сервера в кластере.

SELECT DISTINCT ID, city, address FROM Server NATURAL JOIN Cluster\_c;

Π ID, city, address ( Server>< Cluster\_c)

|  |  |  |
| --- | --- | --- |
| 234.34.452 | Praga | Ve Smeckach, 29 |
| 145.76.469 | Berlin | Hose str, 123 |
| 874.37.324 | San Francisco | Staner str, 35 |
| 145.78.451 | Saint-Petersburg | Ligovckiy pr-t, 64 |
| 132.456.21 | Saint-Petersburg | Sadovaya, 123 |
| 54.454.78 | Praga | Ujezd, 22 |
| 784.41.346 | Praga | Yanski vreshek., 8 |
| 678.245.73 | Berlin | Unter-den-Linden, 22a |
| 64.314.15 | London | Baker str, 22b |
| 13.947.642 | New-York | Ager-roud, 184 |

Запрос с использованием FULL OUTER JOIN.

Вывести логины, подписку, доступную и принадлежащую память пользователей, с бесплатной подпиской.

SELECT login, subs, available, Data\_storage.space\_c

FROM Account FULL JOIN Data\_storage

ON Account.space\_c = Data\_storage.space\_c WHERE subs = 'Free';

|  |  |  |  |
| --- | --- | --- | --- |
| Bravo\_22 | Free | 4.5 | 8 |
| berserk | Free | 12 | 20 |
| Gloria-Melman | Free | 6541.1 | 12000 |

Запрос с использованием RIGHT OUTER JOIN.

Вывести названия облачных хранилищ, директоров их ресурсных центров, распределяемую ими память и ее стоимость для платных подписок.

Select name, director, space\_c, cost

FROM Resource\_center RIGHT JOIN Data\_storage ON cost != 0 AND Resource\_center.city = Data\_storage.city;

|  |  |  |  |
| --- | --- | --- | --- |
| Mega | Jan Becher | 64 | 2 |
| Mega | Jan Becher | 16 | 1 |
| NextCloud | Sergey Khrustin | 12000 | 12 |
| NextCloud | Sergey Khrustin | 39 | 1 |
| OneDrive | Hanz Lyahter | 1020 | 10 |
| OneDrive | Hanz Lyahter | 128 | 3 |
| Azure | Rick Sunches | 400 | 4 |
| Azure | Rick Sunches | 500 | 6 |
| - | - | 20 | 0 |
| - | - | 8 | 0 |

Запрос с использованием LEFT OUTER JOIN.

Вывести ID кластеров и названия фирм, производящих вычисления в них вычисления с временем непрерывной работы от 0 до 4 лет.

Select ID, client, life\_time

FROM Cluster\_c LEFT OUTER JOIN Calculation using(city)

WHERE life\_time BETWEEN 0 and 4;

|  |  |  |
| --- | --- | --- |
| 54.454.78 | Parallels | 2 |
| 54.454.78 | Wrike | 2 |
| 54.454.78 | Luxsoft | 2 |
| 678.245.73 | Vcontacte | 4 |
| 678.245.73 | Wargaming | 4 |
| 678.245.73 | Avito | 4 |
| 132.456.21 | SberTech | 4 |
| 132.456.21 | JetBrains | 4 |

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

Запрос с использованием AVG.

Вывести названия всех фирм, скорость и стоимость вычислений, где скорость вычислений выше средней.

Select client, price, speed

FROM Calculation

WHERE speed >ALL (Select AVG(speed) FROM Calculation);

|  |  |  |
| --- | --- | --- |
| Wrike | 100 | 5.5 |
| SberTech | 782.14 | 3.1 |
| JetBrains | 130.6 | 9.61 |
| SEMrush | 125.66 | 4.8 |

10. Агрегация и GROUP BY - 1 запрос.

Вывести все компании и число их сотрудников.

Select client, COUNT(DISTINCT login) FROM Account GROUP BY client;

|  |  |
| --- | --- |
| SEMrush | 3 |
| JetBrains | 1 |
| Vcontacte | 2 |
| Wrike | 3 |
| Epam | 1 |

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

Вывести директоров ресурсных, числа подконтрольных им узлов и серверовцентров, под управлением которых находится больше 5000 серверов.

Select director, SUM(numb\_nodes), (SUM(numb\_nodes) \* SUM(numb\_servers)) Sum\_nodes

FROM Resource\_center, Cluster\_c WHERE Resource\_center.city = Cluster\_c.city

GROUP BY director HAVING Sum\_nodes>5000 ORDER BY Sum\_nodes;

|  |  |  |
| --- | --- | --- |
| Jan Becher | 33 | 10131 |

12. Коррелированный запрос – 2 запроса разного типа.

Найти города, в которых производят вычисления сразу несколько фирм.

Select DISTINCT A1.city FROM Calculation A1 WHERE city IN (Select A2.city FROM Calculation A2 WHERE A1.client <> A2.client);

|  |
| --- |
| Praga |
| Berlin |
| Saint-Petersburg |
| New-York |

Найти cерийные номера и ID серверов, находящихся под управлением компаниии Apple.

Select DISTINCT serial\_numb, ID\_C FROM Server WHERE EXISTS (Select \* FROM Cluster\_c

WHERE ID\_C = ID and Cluster\_c.city IN (Select Resource\_center.city FROM Resource\_center

WHERE Resource\_center.name = ANY (Select Cloud.name FROM Cloud WHERE company = 'Apple')));

|  |  |
| --- | --- |
| 94165549 | 874.37.324 |
| 28449488 | 874.37.324 |
| 21954544 | 874.37.324 |

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 old предназначенный для поддержания ссылочной целостности.

При изменениях (удалении) данных в одной таблице триггер удаляет данные и в других таблицах, на которые есть ссылка.

create or replace trigger Cluster\_trigger

after delete on Cluster\_c

for each row

begin

delete from Server where ID\_C=:old.ID;

end Cloud\_trigger;

В качестве примера удалим все кластеры, существующие 2 года.

delete from Cluster\_c where life\_time=2;

После операции в таблице не оказывается таких данных.

select \* from Cluster\_c where life\_time=2;

|  |
| --- |
| no data found |

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

Создадим новую таблицу аудита аккаунтов облачных хранилищ, в которую будем записывать все изменения данных (пользователь который вносил изменения (user) , системное время совершения операции (time), операция (operation) и само изменение (hist).

create table Account\_audit ( users varchar2 (30), time Timestamp, operation varchar2 (20), hist varchar2(60));

Этот триггер позволяет проверять, какое действие вызвало его срабатывание и отслеживать изменения кортежей таблицы.

create or replace trigger Account\_trigger

after insert or delete or update on Account

for each row

declare

oper varchar2(20);

begin

if inserting then oper:='insert'; End if;

if deleting then oper:='delete'; End if;

if updating then oper:='update'; End if;

insert into Account\_audit values (user, sysdate, oper,

'oldName:'||:old.login||'newName:'||:new.login);

End Account\_trigger;

В качестве примера удалим все аккаунты, где клиентом является фирма Wrike.

delete from Account where client='Wrike';

Выведем все записи таблицы Account\_audit.

Select \* from Account\_audit;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| USERS | TIME | OPERATION | HIST | USERS |
| ANONYMOUS | 09-DEC-18 10.02.38.000000 PM | delete | oldName:Bravo\_22newName: | ANONYMOUS |
| ANONYMOUS | 09-DEC-18 10.02.38.000000 PM | delete | oldName:SPbSTUnewName: | ANONYMOUS |
| ANONYMOUS | 09-DEC-18 10.02.38.000000 PM | delete | oldName:bremyanewName: | ANONYMOUS |

3. Триггер с When

Отражает изменение данных в таблице по условию When.

Создадим таблицу больших вычислений, в которую будем записывать заказчика, стоимость и затраченные на вычисления ресурсы.

create table BigCalculation (client varchar2(50),

price float,

cosumption int);

create or replace trigger BigCalc\_trigger

after insert or update on Calculation

for each row

when (new.cosumption>20000)

begin

insert into BigCalculation

(client, price, cosumption)

values (:new.client,:new.price,:new.cosumption);

End BigCalc\_trigger;

Обновим данные таблицы так, что у клиента Джетбрейнс на вычисления было затрачено 30000 мегаватт.

update Calculation set cosumption=30000 where client = 'JetBrains';

Выведем все записи таблицы больших вычислений.

select \* from BigCalculation;

|  |  |  |
| --- | --- | --- |
| CLIENT | PRICE | COSUMPTION |
| JetBrains | 130.6 | 30000 |

4. Программирование клиента базы данных

При разработке баз данных операторы языка SQL включаются в программу, составленную на основном языке программирования, в данном случае Java. Препроцессор заменяет операторы SQL на функции основного языка и может компилировать полученный код.

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 переводит последовательно между кортежами таблицы Data\_storage с заданным значением login суммы денег. В случае ошибки или неудачного перевода из какой-либо кортежа, транзакция будет полностью отменена. Таким образом обеспечивается целостность данных.

package javaapplication1;

import java.sql.\*;

import java.util.\*;

public class Main {

static final String JDBC\_DRIVER = "sun.jdbc.odbc.JdbcOdbcDriver";

static final String DATABASE\_URL = "jdbc:odbc:cloud";

static final String USER = "letya999";

static final String PASSWORD = "1111";

public static void main(String[] args) throws ClassNotFoundException, SQLException {

Connection con = null;

Statement st = null;

int space\_c=0,cost\_c=0;

int space\_s=0, cost\_s=0;

Class.forName(JDBC\_DRIVER);

try {

con = DriverManager.getConnection(DATABASE\_URL, USER, PASSWORD);

con.setAutoCommit(false);

st = con.createStatement();

Scanner s = new Scanner(System.in);

String str1 = s.nextLine();

String SQL = "select Account\_Data\_Calc.space\_c, Account\_Data\_Calc.cost from Account\_Data\_Calc where login='"+str1+"'";

ResultSet customer=st.executeQuery(SQL);

while(customer.next()) {

space\_c=customer.getInt(1);

cost\_c=customer.getInt(2);

}

String str2 = s.nextLine();

SQL="select Account\_Data\_Calc.space\_c, Account\_Data\_Calc.cost from Account\_Data\_Calc where login='"+str2+"'";

ResultSet seller=st.executeQuery(SQL);

while(seller.next()) {

space\_s=seller.getInt(1);

cost\_s=seller.getInt(2);

}

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

int money = s.nextInt();

if(money<cost\_c) {

cost\_c=cost\_c-money;

int val = st.executeUpdate("update Data\_storage set cost="+cost\_c+"where space\_c="+space\_c);

cost\_s=cost\_s+money;

val = st.executeUpdate("update Data\_storage set cost="+cost\_s+"where space\_c="+space\_s);

}

con.commit();

} catch (SQLException e) {

e.getSQLState();

e.printStackTrace();

con.rollback();

con.close();

}

}

}

4.2 Вторая транзакция

Для обеспечения параллельности одновременно выполняемых транзакции был выбран режим изоляции SERIALIZABLE, который устанавливается с помощью функции setTransactionIsolation() и позволяет избежать случаев потерянных обновлений, чтения грязных данных, неповторяющегося чтения и фантомной вставки. Этот режим изоляции является самым надежным, однако и самым медленным, поскольку позволяет выполнять транзакции только последовательно, что замедляет работу БД.

В приведенном коде производится ввод логин и пароль пользователя облачного хранилища и печать информации об аккаунте. Затем пользователю предлагается сменить пароль и ввести новый, после его подтверждения изменения вносятся в базу данных. В данном примере выбран именно этот уровень изоляции, поскольку он позволяет избежать эффекта фантомных вставок, который возникнет при выполнении параллельно этой транзакции много раз над одним и тем же кортежем, а также потому, что режим REPEATABLE READ не поддерживается базой данных СУБД oracle 11g XE, а другие режимы является менее надежными.

public static void change\_pass (String login, String password, Connection con)throws SQLException {

String date, subs, client;

try {

con.setTransactionIsolation(8);

con.setAutoCommit(false);

Statement st = con.createStatement();

ResultSet man=st.executeQuery("select \* from Account where password='"+password+"'");

while(man.next()) {

subs = man.getString(3);

date =man.getString(4);

client=man.getString(5);

System.out.println(client+"\t"+login+"\t"+password+"\t"+subs+"\t"+date);

}

Scanner s = new Scanner(System.in);

System.out.println("Enter new password");

password = s.nextLine();

int val = st.executeUpdate("update Account set password='"+password+"' where login='"+login+"'");

System.out.println("For consent, enter ok");

String ch = s.nextLine();

if(ch.equals("ok")) con.commit();

else con.rollback();

} catch (SQLException e) {

e.printStackTrace();

con.rollback();

}

}

Вывод

В ходе написания данной курсовой работы была изучена литература по темам базы данных и системы управления базами данных, также были изучены ключевые особенности языка определения данных ODL, языка структурированных запросов SQL, а также его модификация PL/SQL и программирование клиента базы данных с помощью методов JDBC и Java.

В первой части проделанной работы была создана ODL модель базы данных облачного хранилища, представляющая все типы возможных связей отношений, а также ER-модель, реализующая визуально-графическое представление отношений и связей между ними. 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 заключительной части курсовой работы была разработана программа клиента базы данных на 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 и 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/>