|  |
| --- |
| **Лабораторная работа №4. Реляционная алгебра.** |
| Задание №1.  Настройка SQL\*Plus. Необходимо включить режим ECHO и вывести протокол лаб. работы в файл *<Фамилия студента>.txt*. Этот файл является отчетом о проделанной лаб. работе.  Пример:  Следующий пример включает режим ECHO и ведет файл протокола spool. txt  SQL> set echo on  SQL> spool d:\spool.txt  Задание №2.  --  -- Copyright (c) Oracle Corporation 1988, 2000. All Rights Reserved.  --  -- NAME  -- demobld.sql  --  -- DESCRIPTION  -- This script creates the SQL\*Plus demonstration tables in the  -- current schema. It should be STARTed by each user wishing to  -- access the tables. To remove the tables use the demodrop.sql  -- script.  --  -- USAGE  -- From within SQL\*Plus, enter:  -- START demobld.sql    SET TERMOUT ON  PROMPT Building demonstration TABLES. Please wait.  SET TERMOUT OFF    DROP TABLE EMP;  DROP TABLE DEPT;  DROP TABLE DEPT1;  DROP TABLE BONUS;  DROP TABLE SALGRADE;  DROP TABLE DUMMY;    CREATE TABLE EMP  (EMPNO NUMBER(4) NOT NULL,  ENAME VARCHAR2(10),  JOB VARCHAR2(9),  MGR NUMBER(4),  HIREDATE DATE,  SAL NUMBER(7, 2),  COMM NUMBER(7, 2),  DEPTNO NUMBER(2));    INSERT INTO EMP VALUES  (7369, 'SMITH', 'CLERK', 7902,  TO\_DATE('17-12-1980', 'DD-MM-YYYY'), 800, NULL, 20);  INSERT INTO EMP VALUES  (7369, 'SMITH', 'CLERK', 7902,  TO\_DATE('17-12-1980', 'DD-MM-YYYY'), 800, NULL, 20);  INSERT INTO EMP VALUES  (7499, 'ALLEN', 'SALESMAN', 7698,  TO\_DATE('20-02-1981', 'DD-MM-YYYY'), 1600, 300, 30);  INSERT INTO EMP VALUES  (7521, 'WARD', 'SALESMAN', 7698,  TO\_DATE('22-02-1981', 'DD-MM-YYYY'), 1250, 500, 30);  INSERT INTO EMP VALUES  (7566, 'JONES', 'MANAGER', 7839,  TO\_DATE('2-4-1981', 'DD-MM-YYYY'), 2975, NULL, 20);  INSERT INTO EMP VALUES  (7654, 'MARTIN', 'SALESMAN', 7698,  TO\_DATE('28-9-1981', 'DD-MM-YYYY'), 1250, 1400, 30);  INSERT INTO EMP VALUES  (7698, 'BLAKE', 'MANAGER', 7839,  TO\_DATE('1-5-1981', 'DD-MM-YYYY'), 2850, NULL, 30);  INSERT INTO EMP VALUES  (7782, 'CLARK', 'MANAGER', 7839,  TO\_DATE('9-6-1981', 'DD-MM-YYYY'), 2450, NULL, 10);  INSERT INTO EMP VALUES  (7788, 'SCOTT', 'ANALYST', 7566,  TO\_DATE('09-12-1982', 'DD-MM-YYYY'), 3000, NULL, 20);  INSERT INTO EMP VALUES  (7839, 'KING', 'PRESIDENT', NULL,  TO\_DATE('17-11-1981', 'DD-MM-YYYY'), 5000, NULL, 10);  INSERT INTO EMP VALUES  (7844, 'TURNER', 'SALESMAN', 7698,  TO\_DATE('8-9-1981', 'DD-MM-YYYY'), 1500, 0, 30);  INSERT INTO EMP VALUES  (7876, 'ADAMS', 'CLERK', 7788,  TO\_DATE('12-1-1983', 'DD-MM-YYYY'), 1100, NULL, 20);  INSERT INTO EMP VALUES  (7900, 'JAMES', 'CLERK', 7698,  TO\_DATE('3-12-1981', 'DD-MM-YYYY'), 950, NULL, 30);  INSERT INTO EMP VALUES  (7902, 'FORD', 'ANALYST', 7566,  TO\_DATE('3-12-1981', 'DD-MM-YYYY'), 3000, NULL, 20);  INSERT INTO EMP VALUES  (7934, 'MILLER', 'CLERK', 7782,  TO\_DATE('23-1-1982', 'DD-MM-YYYY'), 1300, NULL, 10);  INSERT INTO EMP VALUES  (7934, 'MILLER', 'CLERK', 7782,  TO\_DATE('23-1-1982', 'DD-MM-YYYY'), 1300, NULL, 10);  CREATE TABLE DEPT  (DEPTNO NUMBER(2),  DNAME VARCHAR2(14),  LOC VARCHAR2(13) );    INSERT INTO DEPT VALUES (10, 'ACCOUNTING', 'NEW YORK');  INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DALLAS');  INSERT INTO DEPT VALUES (30, 'SALES', 'CHICAGO');  INSERT INTO DEPT VALUES (40, 'OPERATIONS', 'BOSTON');    CREATE TABLE BONUS  (ENAME VARCHAR2(10),  JOB VARCHAR2(9),  SAL NUMBER,  COMM NUMBER);    CREATE TABLE SALGRADE  (GRADE NUMBER,  LOSAL NUMBER,  HISAL NUMBER);    INSERT INTO SALGRADE VALUES (1, 700, 1200);  INSERT INTO SALGRADE VALUES (2, 1201, 1400);  INSERT INTO SALGRADE VALUES (3, 1401, 2000);  INSERT INTO SALGRADE VALUES (4, 2001, 3000);  INSERT INTO SALGRADE VALUES (5, 3001, 9999);    CREATE TABLE DUMMY  (DUMMY NUMBER);  INSERT INTO DUMMY VALUES (0);  COMMIT;  SET TERMOUT ON  PROMPT DEMONSRTRATION TABLE BUILD IS COMPLETE  **Реляционная алгебра**  Основные операторы   — само отношение А (отношение здесь синонимично с таблицей и предикатом) является выражением реляционной алгебры, более того, так как это алгебра, любое выражение реляционной алгебры возвращает отношение (свойство замыкания операторов)  **Selection (выборка; ограничение)**   — selection (выборка; ограничение), A — отношение (предикат, таблица),  – булева формула, по которой происходит отбор строк (кортежей, записей, etc)  Selection является по сути горизонтальным фильтром строк, т.е., можно представить, что мы идем по каждой строке и оставляем только те, что удовлетворяют условию . Простой пример для наглядности: C:\Users\студент\Downloads\image.png  **Projection (проекция)**   — projection (проекция) на атрибуты A, B, …. Возвращает таблицу, в которой остаются только колонки (атрибуты) A, B, …. Простой пример ниже. По сути является фильтром по атрибутам т.е. это в некотором смысле вертикальный фильтр. C:\Users\студент\Downloads\image (1).png  **Переименование**   — переименовывает колонку a в b в отношении A (атрибут, аргумент предиката, etc); два чая тому господину, который покажет, что алгебра строго более выразима с оператором переименования (нужно привести пример запроса, который не выразим без этого оператора, но выразим с )  **Декартово произведение**   — Декартово произведение двух отношений, большое отношение из всевозможных сочетаний строк в A и B.  C:\Users\студент\Downloads\image (2).png  **Операции над множествами**  Реляционная алгебра является расширением классического набора операторов над множествами (отношение — это множество упорядоченных кортежей; заметьте, что это совсем не равно упорядоченному множеству кортежей). Пусть у нас есть таблица StudentMark(Name, Mark, Subject, Date) – тогда кортеж (Вася, 5, Информатика, 05.10.2010) является упорядоченным – сначала строка Name на первой (ок, или нулевой) позиции, целое число на второй, строка на третьей и дата на четвертой. При этом сами упорядоченные кортежи (Name, Mark, Subject, Date) не упорядочены «внутри» отношения.  **Объединение**  C:\Users\студент\Downloads\image (3).png   — объединение всех строк в A и B, ограничение — одинаковые аттрибуты   **Пересечение**  **C:\Users\студент\Downloads\image (4).png**   — пересечение строк, такое же ограничение  **Разница множеств**   — B минус A, все строки, что присутствуют в B, но не присутствуют в A, такое же ограничение  C:\Users\студент\Downloads\image (5).png  (B\A; A — слева, B — справа)  **Вспомогательные операторы**  — join (соединение); join соединяет две записи таблиц A и B, при условии, что для этих двух записей выполнено условие φ.  C:\Users\студент\Downloads\image (6).png  *Реляционная алгебра* была представлена E. F. Codd в 1972 году. Она состоит из множества операций над отношениями:  **ВЫБОРКА(SELECT) (σ):** извлечь *кортежи* из отношения, которые удовлетворяют заданным условиям. Пусть *R* - таблица, содержащая атрибут *A*. σA=a(R) = {t ∈ R ∣ t(A) = a} где t обозначает кортеж *R* и t(A) обозначает значение атрибута *A* кортежа t. где t(X) обозначает значение атрибута X кортежа t.  Операция выборки работает с одним отношением https://function-x.ru/sets/db001.gif и определяет результирующее отношение R, которое содержит только те кортежи (или строки, или записи), отношения https://function-x.ru/sets/db001.gif, которые удовлетворяют заданному условию (предикату P).  Таким образом, операция выборки - унарная операция - и записывается следующим образом:  https://function-x.ru/sets/db002.gif,  где P - предикат (логическое условие).  SELECT \* from EMP WHERE SAL>2500;  **ПРОЕКЦИЯ(PROJECT) (π):** извлечь заданные *атрибуты* (колонки) из отношения. Пусть R отношение, содержащее атрибут X. πX(R) = {t(X) ∣ t ∈ R},  Из исходного отношения выбираем только столбцы ENAME,DEPTNO и видим, что строки со значениями - первая и третья - идентичны. Исключаем дубликат (за это отвечает ключевое слово DISTINCT в SQL-запросе, которое говорит, что нужно выбрать только уникальные записи) и получаем следующее новое отношение, в котором два атрибута и две строки (записи):  SELECT DISTINCT ENAME,DEPTNO from EMP;  SELECT DISTINCT DEPTNO, ENAME from EMP;  SELECT DISTINCT DEPTNO from EMP;  **ПРОИЗВЕДЕНИЕ(PRODUCT) (×):** построить декартово произведение двух отношений. Пусть R - таблица, со степенью k1 и пусть S таблица со степенью k2. R × S - это множество всех k1 + k2 - кортежей, где первыми являются k1 элементы кортежа R и где последними являются k2 элементы кортежа S.  **Операция декартова произведения**  Операция декартова произведения (https://function-x.ru/sets/db004.gif) определяет новое отношение R, которое является результатом конкатенации каждого кортежа отношения R1 с каждым кортежем отношения R2.  Установим, что получится в результате выполнения этой операции реляционной алгебры и соответствующего ей запроса SQL. Даны два отношения R3 и R4:  В новом отношении должны присутствовать все атрибуты (столбцы) двух отношений. Сначала первая строка отношения R1 сцепляется с каждой из двух строк отношения R2, затем вторая строка отношения R1, затем третья. В результате должно получиться 4 Х 5 = 20 кортежей (строк). Получаем такое новое отношение:  SELECT \* from DEPT;  SELECT \* from SALGRADE;  SELECT \* from DEPT, SALGRADE;  SELECT \* from SALGRADE, DEPT;  **ОБЪЕДИНЕНИЕ(UNION) (∪):** построить теоретико-множественное объединение двух таблиц. Даны таблицы R и S (обе должны иметь одинаковую степень), объединение R ∪ S - это множество кортежей, принадлежащих R или S или обоим.  **Операция объединения**  Результатом объединения двух множеств (отношений) А и В (https://function-x.ru/sets/s020.gif) будет такое множество (отношение) С, которое включает в себя те и только те элементы, которые есть или во множестве А или во множестве В. Говоря упрощённо, все элементы множества А и множества В, за исключением дубликатов, образующихся за счёт того, что некоторые элементы есть и в первом, и во втором множестве. Операция объединения реляционной алгебры идентична операции **[объединения множеств](https://function-x.ru/sets1.html" \t "_blank)**. Теперь посмотрим, что получится в результате выполнения этой операции реляционной алгебры и соответствующего ей запроса SQL. Теперь даны два отношения, так как операция объединения - бинарная операция:  Объединяем строки первого и второго отношения и видим, что идентичные строки включаются в новое отношение только один раз. Важно следующее: операция объединения может быть выполнена только тогда, когда два отношения обладают одинаковым числом и названиями атрибутов (столбцов), или, говоря формально, совместимы по объединению.  SELECT DEPTNO,DNAME, LOC from DEPT;  SELECT ENAME,DEPTNO from EMP UNION SELECT DEPTNO,DNAME, LOC from DEPT;  SELECT DEPTNO from EMP UNION SELECT DEPTNO from DEPT;  CREATE TABLE DEPT1  (DEPTNO NUMBER(2),  DNAME VARCHAR2(14),  LOC VARCHAR2(13) );    INSERT INTO DEPT1 VALUES (10, 'ACCOUNTING', 'NEW YORK');  INSERT INTO DEPT1 VALUES (20, 'RESEARCH', 'DALLAS');  INSERT INTO DEPT1 VALUES (10, 'SALES1', 'CHICAGO');  INSERT INTO DEPT1 VALUES (49, 'OPERATIONS1', 'BOSTON');    SELECT \* from DEPT;  SELECT \* from DEPT1;  SELECT DEPTNO,DNAME, LOC from DEPT UNION SELECT DEPTNO,DNAME, LOC from DEPT1;  SELECT DEPTNO,DNAME from DEPT UNION SELECT DEPTNO,DNAME from DEPT1;  **ПЕРЕСЕЧЕНИЕ(INTERSECT) (∩):** построить теоретико-множественное пересечение двух таблиц. Даны таблицы R и S, R ∪ S - это множество кортежей, принадлежащих R и S. Опять необходимо, чтобы R и S имели одинаковую степень.  Результатом пересечения двух множеств (отношений) А и В (https://function-x.ru/sets/s028.gif) будет такое множество (отношение) С, которое включает в себя те и только те элементы, которые есть и во множестве А, и во множестве В. Операция пересечения реляционной алгебры идентична операции **[пересечения множеств, которая также описана в материале "Множества и операции над множествами"](https://function-x.ru/sets1.html" \t "_blank)**.  В некоторых диалектах SQL отсутствует ключевое слово INTERSECT. Поэтому, например, в MySQL и других, операция пересечения множеств может реализована **[с применением предиката EXISTS](https://function-x.ru/sql_intersect_except.html" \t "_blank)**.  SELECT DEPTNO,DNAME from DEPT INTERSECT SELECT DEPTNO,DNAME from DEPT1;  **ВЫЧИТАНИЕ(DIFFERENCE) (− или ∖):** построить множество различий двух таблиц. Пусть R и S опять две таблицы с одинаковой степенью. R - S - это множество кортежей R,не принадлежащих S.  Разность двух отношений R1 и R2 (https://function-x.ru/sets/db003.gif) состоит из кортежей (или записей, или строк), которые имеются в отношении R1, но отсутствуют в отношении R2. Отношения R1 и R2 должны быть совместимы по объединению. Операция разности реляционной алгебры идентична операции **[разности множеств](https://function-x.ru/sets1.html" \t "_blank)**.  SELECT DEPTNO,DNAME from DEPT MINUS SELECT DEPTNO,DNAME from DEPT1;  SELECT DEPTNO,DNAME from DEPT1 MINUS SELECT DEPTNO,DNAME from DEPT;  **СОЕДИНЕНИЕ(JOIN) (∏):** соединить две таблицы по их общим атрибутам. Пусть R будет таблицей с атрибутами A,B и C и пусть S будет таблицей с атрибутами C,D и E. Есть один атрибут, общий для обоих отношений, атрибут C. R ∏ S = πR.A,R.B,R.C,S.D,S.E(σR.C=S.C(R × S)). Что же здесь происходит? Во-первых, вычисляется декартово произведение R × S. Затем, выбираются те кортежи, чьи значения общего атрибута C эквивалентны (σR.C = S.C). Теперь мы имеем таблицу, которая содержит атрибут C дважды и мы исправим это, выбросив повторяющуюся колонку.  SELECT D.DEPTNO, D.DNAME from DEPT D JOIN DEPT1 T ON (D.DEPTNO=T.DEPTNO);  SELECT D.DEPTNO, D.DNAME from DEPT D JOIN DEPT1 T ON (D.DEPTNO=T.DEPTNO);  **Операция тета-соединения**  В результате этой операции получается отношение, которое содержит кортежи из декартова произведения отношений R1 и R2 удовлетворяющие предикату Р. Значением предиката Р может быть один из операторов сравнения (<, <=, >, >=, = или !=).  SELECT \* from DEPT, EMP WHERE SAL>3000;  **Пример 2-2. Внутреннее соединение**  Давайте посмотрим на таблицы, которые получаются в результате шагов, необходимых для объединения. Пусть даны следующие две таблицы:  R A | B | C S C | D | E  ---+---+--- ---+---+---  1 | 2 | 3 3 | a | b  4 | 5 | 6 6 | c | d  7 | 8 | 9    Во-первых, мы вычислим декартово произведение R × S и получим:  R x S A | B | R.C | S.C | D | E  ---+---+-----+-----+---+---  1 | 2 | 3 | 3 | a | b  1 | 2 | 3 | 6 | c | d  4 | 5 | 6 | 3 | a | b  4 | 5 | 6 | 6 | c | d  7 | 8 | 9 | 3 | a | b  7 | 8 | 9 | 6 | c | d    После выборки σR.C=S.C(R × S) получим:  A | B | R.C | S.C | D | E  ---+---+-----+-----+---+---  1 | 2 | 3 | 3 | a | b  4 | 5 | 6 | 6 | c | d    Удалить повторяющуюся колонку S.C можно с помощью проекции следующей операцией: πR.A,R.B,R.C,S.D,S.E(σR.C=S.C(R × S)) и получим:  A | B | C | D | E  ---+---+---+---+---  1 | 2 | 3 | a | b  4 | 5 | 6 | c | d     * ДЕЛЕНИЕ(DIVIDE) (÷): Пусть R будет таблицей с атрибутами A, B, C, и D и пусть S будет таблицей с атрибутами C и D. Мы можем определить деление как: R ÷ S = {t ∣ ∀ ts ∈ S ∃ tr ∈ R так, что tr(A,B)=t∧tr(C,D)=ts} где tr(x,y) означает кортеж таблицы R, который состоит только из элементов x и y. Заметим, что кортеж t состоит только из элементов A и B отношения R.   Зададим следующие таблицы  R A | B | C | D S C | D  ---+---+---+--- ---+---  a | b | c | d c | d  a | b | e | f e | f  b | c | e | f  e | d | c | d  e | d | e | f  a | b | d | e    R ÷ S получается  A | B  ---+---  a | b  e | d    SQL> spool off |