Курсы
«Современные технологии программирования»

Базовый SQL, особенности MS-SQL и Oracle

А. Тищенко
elta@list.ru
2008 г.

Соглашение об авторских правах

Язык SQL

Стандартизованный метод доступа к реляционной базе данных и манипулирования хранящимися в ней данными

Стандарты ANSI:

- 1986
- 1989
- 1992 SQL92 (SQL2)
- 1999 SQL99 (SQL3)
- 2003 SQL2003

Выделяют три подмножества языка SQL

- Язык манипулирования данными (DML), например, SELECT, INSERT, UPDATE, DELETE
- Язык определения данных (DDL), например, CREATE, ALTER, DROP
- Язык управления данными (DCL) например, GRANT и REVOKE

Язык определения данных

Создание таблицы

Команда создания таблицы:

CREATE TABLE имя_таблицы
({<столбец>|<ограничение_на_таблицу>}
[,{<столбец>|<ограничение_на_таблицу>}]
)
<столбец>::=имя_столбца тип_данных [DEFAULT выражение]
[<ограничение_на_столбец>]

При создании таблицы дополнительно могут указываться свойства хранения.

Определение ограничений целостности будет рассмотрено далее.

Пример создания таблицы:

```
CREATE TABLE dept2
    (deptno NUMBER(3),
     dname VARCHAR2(10),
     loc VARCHAR2(13));
```

DESC[RIBE] имя_таблицы – команда SQL*PLUS, возвращает описание таблицы

```
DESCRIBE dept2
```

Особенности простейшего определения таблицы в Oracle и SQL Server

`SELECT TABLE_NAME FROM USER_TABLES` – просмотр всех таблиц пользователя

Oracle:

- Полное имя [имя_пользователя.]имя_таблицы
- Позволяет определить таблицу через запрос

  Create table имя_таблицы

  as имя_запроса

Изменение таблицы в Oracle

Возможные действия:

- добавление (**ADD**) модификация (**MODIFY**), удаление (**DROP**) столбцов;
- добавление, модификация и удаление ограничений;
- управление выделенной для таблицы памятью.

Команда изменения таблицы:

ALTER TABLE имя_таблицы
[ADD (<столбец>|<ограничение_уровня_таблицы>
{, <столбец>|<ограничение_уровня_таблицы>})]
[MODIFY (<столбец> {,< столбец>})]
[DROP ограничение|COLUMN имя_столбца]

Пример добавления столбца:

```
ALTER TABLE dept2
ADD (mgr number(4));
```

Простейшие типы данных

	Oracle	SQL Server
Строка переменной длины	VARCHAR2(размер)	VARCHAR[(размер)]
Строка фиксированной длины	CHAR(размер)	CHAR[(размер)]
Числовые	NUMBER(p,s)	NUMERIC(p,s) DECIMAL(p[,s]) INT FLOAT[(n)]
Временные	DATE TIMESTAMP INTERVAL YEAR TO MONTH INTERVAL DAY TO SECOND	DATETIME
Денежные		MONEY, SMALLMONEY

Модификация столбца

Модифицировать можно тип данных столбца, размер и значение по умолчанию.
Изменить тип данных или уменьшить размер можно, если столбец содержит только NULL-значения или таблица не содержит строк.
Можно преобразовать столбец типа CHAR в столбец типа VARCHAR2 и наоборот, если он содержит NULL-значения или если не изменяется размер.
Изменение значения по умолчанию отражается только на будущих вставках данных.

Пример модификации столбца:

```
ALTER TABLE dept2
MODIFY (dname VARCHAR2(15));
```

Удаление столбца

Пример удаления столбца:

```
ALTER TABLE dept2
DROP COLUMN mgr;
```

Командой ALTER TABLE DROP COLUMN столбец удаляется сразу же. Это может занять много времени, если столбец большой. В таком случае столбец можно пометить как неиспльзуемый и удалить его позже.

ALTER TABLE имя_таблицы
SET UNUSED (имя столбца)|COLUMN имя_столбца;
ALTER TABLE имя_таблицы
DROP UNUSED COLUMNS;

Пример удаления столбца с помощью фразы UNUSED:

```
ALTER TABLE dept2
SET UNUSED (dname);
DESC dept2
ALTER TABLE dept2
DROP UNUSED COLUMNS;
```

Неиспользуемые (UNUSED) столбцы

Неиспользуемые (UNUSED) столбцы рассматриваются как удаленные, даже если их данные еще не удалены.
Список неиспользуемых столбцов можно просмотреть в представлении `USER_UNUSED_COL_TABS`

Удаление таблицы

Команда удаления таблицы:

DROP TABLE имя_таблицы

Пример удаления таблицы:

```
DROP TABLE dept2;
```

- В Oracle есть параметр CASCADE CONSTRAINTS, удаляет все ограничения ссылочной целостности

Задания

Работа в схеме HR/HR

1. Создайте таблицу EMP

Имя столбца	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Тип данных	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Длина	7	25	25	7

2. Увеличьте длину столбца LAST_NAME
3. Добавьте столбец JOB_ID типа VARCHAR длины 15
4. Удалите столбец FIRST_NAME
5. Пометьте столбец DEPT_ID как неиспользуемый
6. Посмотрите описание таблицы EMP
7. Удалите неиспользуемый столбец
8. Посмотрите описание таблицы EMP
9. Удалите таблицу EMP

Язык манипулирования данными

Вставка данных в таблицу

Команда INSERT позволяет вставлять строки в таблицу.

Синтаксис команды INSERT:

INSERT INTO имя_таблицы|имя_представления
[(столбец {,столбец})]
[VALUES (значение {,значение})]|подзапрос

Пример вставки строки:

```
INSERT INTO departments (department_id,
  department_name, manager_id, location_id)
VALUES (310, 'Public Relations', 100, 1700);
```

Если столбец в списке пропущен, то автоматически вставляется NULL-значение.

Изменение данных в таблице

Синтаксис команды изменения данных:

UPDATE имя_таблицы|имя_представления [псевдоним]
SET столбец=выражение {,столбец=выражение}
[WHERE условие];

или

UPDATE имя_таблицы|или_представления [псевдоним]
SET (столбец {,столбец}) = (подзапрос)
[WHERE условие]

Пример изменения данных:

```
UPDATE employees
SET department_id=70
WHERE employee_id=113;
```

Удаление данных из таблицы

Для удаления одной или нескольких строк используется команда DELETE:

DELETE [FROM] имя_таблицы|имя_представления [псевдоним]
[WHERE условие];

Пример удаления строк:

```
DELETE FROM employees
WHERE employee_id=133;
```

Если фраза WHERE отсутствует, будут удалены все строки.

Если значение параметра SQL*PLUS AUTOCOMMIT равняется OFF, то можно восстановить данные, удаленные командой DELETE.

Команда TRUNCATE удаляет все строки из таблицы, при этом она работает быстрее чем DELETE, но ее невозможно откатить.

Синтаксис команды TRUNCATE:

TRUNCATE TABLE имя_таблицы;

Задания

1. Создайте таблицу MY_EMPLOYEE

NAME	TYPE
ID	NUMBER(4)
LAST_NAME	VARCHAR2(25)
FIRST_NAME	VARCHAR2(25)
USERID	VARCHAR2(8)
SALARY	NUMBER(9,2)

2. Добавьте в нее строки, не перечисляя столбцы в команде INSERT.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	Dancs	Betty	bdancs	860

Задания

3. Добавьте в таблицу еще две строки, теперь перечисляя столбцы в команде INSERT.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750

4. Проверьте данные в таблице.
5. Измените last_name у работника номер 3 на Drexler.
6. Измените зарплату на 1000 у всех работников с зарплатой меньше 900.
7. Проверьте данные в таблице.
8. Удалите Betty Dancs из таблицы MY_EMPLOYEE.
9. Проверьте данные в таблице.
10. Удалите все данные из таблицы с помощью команды TRUNCATE.
11. Проверьте данные в таблице.

Выборка данных

Простейший вариант команды выборки данных:

SELECT *| {[ALL|DISTINCT] столбец [псевдоним], ... }
FROM {таблица, ... }
[WHERE условие(я)]
[ORDER BY {столбец|выражение, ... } [ASC|DESC]]

Символ * во фразе SELECT означает выбор всех столбцов
Ключевое слово DISTINCT позволяет избежать дублирования строк
Во фразе FROM задается список таблиц, из которых производится выборка
Фраза WHERE задает условия отбора строк
Фраза ORDER BY упорядочивает строки по возрастанию(ASC, сортировка по умолчанию) или по убыванию(DESC)

Использование команды SELECT

Пример выборки данных:

```
SELECT DISTINCT department_id
FROM employees;
```

Пример использования арифметических выражений и задания псевдонимов столбцов:

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM employees;
```

Оператор конкатенации || соединяет столбцы и символьные строки, например:

```
SELECT first_name||' '||last_name employee
FROM employees;
```

NULL это универсальное (не зависящее от типа данных) значение, показывающее, что истинное значение неизвестно. Любые алгебраические операции с операндом null должны давать также неопределенное значение null.

```
SELECT last_name, 12*salary*commission_pct "Annual
Commission" FROM employees;
```

Фраза WHERE (1/5)

Простейшие условия во фразе WHERE – это условия сравнения:

=	равно
>	больше чем
>=	больше или равно
<	меньше чем
<=	меньше или равно
<>,!=, ^=	не равно

Отличия в условиях сравнения в MS SQL Server:
отсутствует ^=, но есть !<(не меньше чем), !>(не больше чем)
При сравнении символьные константы и значения дат заключаются в одинарные кавычки. Символьные строки чувствительны к регистру, а даты к формату, например:

```
SELECT last_name, job_id, department_id
FROM employees
WHERE last_name ='WHALEN';
```
(сравните `'Whalen'`)

Фраза WHERE (2/5)

Другие условия сравнения:

BETWEEN <выражение1> **AND** <выражение2> между двумя значениями (включительно), например:

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 2500 AND 3500;
```

IN (список) наличие в списке; элементы списка перечисляются через запятую, например:

```
SELECT employee_id, last_name, salary,
  manager_id
FROM employees
WHERE manager_id IN (100, 101,201);
```

Фраза WHERE (3/5)

Оператор **LIKE** проверяет соответствие шаблону

Оператор LIKE позволяет искать значения по вхождению в них символа. Для задания шаблона поиска применяют два символа:

% означает любую последовательность символов, в том числе пустую;

_ задает точно один символ.

Шаблон есть текстовая константа и потому записывается в апострофах, например, '_WE%'

```
SELECT first_name
FROM employees
WHERE first_name LIKE 'S%';
```

Оператор **IS NULL** является ли значение NULL.

```
SELECT last_name, manager_id
FROM employees
WHERE manager_id IS NULL;
```

Фраза WHERE (4/5)

Логические условия:

- **AND / OR** логические связки для объединения условий
- **NOT** отрицание условия

При обработке данных с неопределенными значениями необходимо пользоваться трехзначной логикой:

AND	F	T	U
F	F	F	F
T	F	T	U
U	F	U	U

OR	F	T	U
F	F	T	U
T	T	T	T
U	U	T	U

NOT	
F	T
T	F
U	U

Пример логической связки во фразе WHERE

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary>=10000
AND job_id LIKE '%MAN%';
```

Фраза WHERE (5/5)

Порядок выполнения операторов:

1. Арифметические операторы
2. Оператор конкатенации
3. Условия сравнения
4. IS [NOT] NULL, LIKE, [NOT] IN
5. [NOT] BETWEEN
6. Логическое условие NOT
7. Логическое условие AND
8. Логическое условие OR

Изменить порядок выполнения операторов можно с помощью скобок: ( )

Фраза ORDER BY

Примеры использования фразы ORDER BY:

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date;

SELECT employee_id, last_name, salary*12 annsal
FROM employees
ORDER BY annsal;

SELECT last_name, department_id, salary
FROM employees
ORDER BY department_id, salary DESC;
```

Задания

1. Создайте запрос, который выводит last_name и salary всех работников, зарабатывающих более 12000$.
2. Создайте запрос, который отображает last_name и salary всех работников, чья зарплата не находится в интервале 5000$ и 12000$ Назовите столбцы Employee и Monthly Salary, соответственно.
3. Отобразите last_name и department_id для всех работников отделов 20 и 50, упорядочьте по first_name.
4. Отобразите last_name и job_id всех работников, которые не имеют менеджера.
5. Отобразите last_name для всех работников, у которых 3-я буква в last_name равняется a.

Функции Oracle(1/4)

- Символьные функции

1) Функции манипуляции регистром

Функция	Результат
`LOWER('SQL Course')`	`sql course`
`UPPER('SQL Course')`	`SQL COURSE`
`INITCAP('SQL Course')`	`Sql Course`

Пример работы с функциями манипулирования регистром:

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE last_name='higgins';
строки не выбраны
```

А так?

```
SELECT employee_id, last_name, department_id
FROM employees
WHERE LOWER(last_name)='higgins';
```

Функции Oracle(2/4)

2)Функции манипулирования символьными строками

Функция	Результат
`CONCAT('Hello', 'World')`	`HelloWorld`
`SUBSTR('HelloWorld',1,5)`	`Hello`
`LENGTH('HelloWorld')`	`10`
`INSTR('HelloWorld','W')`	`6`
`LPAD(salary,10,'*')`	`*****24000`
`RPAD(salary,10,'*')`	`24000*****`
`TRIM('H' FROM 'HelloWorld')`	`elloWorld`

Функции Oracle(3/4)

- Числовые функции

ABS возвращает абсолютную величину числа

BITAND побитовое AND

CEIL возвращает значение, округленное до ближайшего большего целого

FLOOR возвращает значение, округленное до ближайшего меньшего целого

MOD остаток от деления

SIGN возвращает -1, если аргумент отрицательный, 0 – если аргумент равен 0 и 1 – если аргумент положительный

ROUND округление по математическим правилам

TRUNC округление путем отбрасывания

Функции Oracle(4/4)

- Работы с датами
- Преобразования типов данных
- Общие

Очень часто по смыслу задачи неопределенное значение можно при вычислениях заменить каким-то определенным значением.

Это позволяет сделать функция NVL, имеющая формат:

NVL(имя, значение).

Только в Oracle есть функция

Dump(выражение)-возвращает внутреннее представление выражения

```
select dump('abc') from dual;
```

Соединения

Если в SQL-запросе необходимо получить данные из двух или более таблиц, используют **соединения таблиц**.

ANSI синтаксис соединений(поддерживается в Oracle начиная с 9i):

SELECT * | {[DISTINCT] столбец [псевдоним], ... }
FROM **таблица1**
[CROSS JOIN таблица2] |
[NATURAL JOIN таблица2] |
[JOIN таблица2 USING (имя_столбца)] |
[JOIN таблица2 ON (условие)] |
[LEFT | RIGHT | FULL OUTER JOIN таблица2 ON (условие)];

CROSS JOIN задает декартово произведение таблиц

NATURAL JOIN соединяет две таблицы на основе являющихся для них общими столбцов ключа

USING позволяет указать имя столбца, общего для обеих таблиц

ON (условие) задание условия соединения таблиц, может быть по равенству и не по равенству

LEFT|RIGHT|FULL OUTER JOIN – внешнее соединение. Позволяет извлечь строки из одной таблицы, которые не совпадают со строками из другой таблицы

Примеры использования соединений

- CROSS JOIN декартово произведение таблиц:

```
SELECT last_name, department_name
FROM employees
CROSS JOIN departments;
```

- NATURAL JOIN (соединяет две таблицы на основе являющихся для них общими столбцов ключа):

```
SELECT department_id, department_name, location_id, city
FROM departments
NATURAL JOIN locations;
```

- Фраза USING позволяет указать имя столбца, общего для обеих таблиц:

```
SELECT l.city, d.department_name
FROM locations l JOIN departments d USING (location_id)
WHERE location_id=1400;
```

- Фраза ON задает условие соединения таблиц:

```
SELECT e.employee_id, e.last_name, e.department_id,
   d.department_id, d.location_id
FROM employees e JOIN departments d
ON (e.department_id=d.department_id);
```

Примеры использования внешних соединений

- LEFT OUTER JOIN – извлекает записи, удовлетворяющие условию соединения, и те, для которых нет совпадения в правой таблице

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e
LEFT OUTER JOIN departments d
ON (e.department_id=d.department_id);
```

- RIGHT OUTER JOIN - извлекает записи, удовлетворяющие условию соединения, и те, для которых нет совпадения в левой таблице

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e
RIGHT OUTER JOIN departments d
ON (e.department_id=d.department_id);
```

- FULL OUTER JOIN - извлекает записи, удовлетворяющие условию соединения, и те, для которых нет совпадения в левой таблице и правой таблице

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e
FULL OUTER JOIN departments d
ON (e.department_id=d.department_id);
```

Старые обозначения соединений в Oracle

SELECT столбцы
FROM таблица1, таблица
WHERE условие_соединения

Могут быть по равенству и не по равенству

Пример соединения по равенству:

```
SELECT e.employee_id, e.last_name,
e.department_id, d.department_id, d.location_id
FROM employees e, departments d
WHERE e.department_id=d.department_id;
```

Пример соединения не по равенству:

```
SELECT e.ename, e.sal, sg.grade
FROM emp e, salgrade sg
WHERE e.sal BETWEEN sg.losal AND sg.hisal;
```

Старые обозначения внешних соединений в Oracle

Внешнее соединение **(+)** проставляется на той стороне, где могут отсутствовать данные:

таблица1.имя_столбца=таблица2.имя_столбца(+)

таблица1.имя_столбца(+)=таблица2.имя_столбца

Примеры внешних соединений:

```
SELECT e.last_name, e.department_id, d.department_name
FROM employees e, departments d
WHERE e.department_id(+)=d.department_id;

SELECT e.last_name, e.department_id, d.department_name
FROM employees e, departments d
WHERE e.department_id=d.department_id(+);
```

Если строки не имеющие пары могут иметься в двух таблицах сразу, используйте объединение UNION.

Соединение таблицы с собой

Если таблица содержит иерархическую структуру, то могут использоваться ее соединения с собой. Чтобы выполнить такое соединение, вводят два разных псевдонима во фразе FROM

Пример соединения таблицы с собой в старом синтаксисе:

```
SELECT worker.last_name||' works for
  '||manager.last_name
FROM employees worker, employees manager
WHERE worker.manager_id=manager.employee_id;
```

Пример соединения таблицы с собой в новом синтаксисе:

```
SELECT e.last_name emp, m.last_name mgr
FROM employees e JOIN employees m
ON (e.manager_id=m.employee_id);
```

Задания

1. Напишите запрос, отображающий last_name, department_number и department_name для всех работников.
2. Напишите запрос, отображающий last_name, department_name, location_id и city для всех работников, получающих комиссонные.
3. Отобразите last_name, employee_id работника и last_name и id менеджера.
4. Модифицируйте запрос номер 3 так, чтобы он отображал всех работников, включая работника King, который не имеет менеджера.

Фраза GROUP BY

SELECT * | {[DISTINCT] столбец [псевдоним], ... }
FROM {таблица, ... }
WHERE критерии_отбора_для_всей_таблицы
GROUP BY столбец1, [столбец2, ...]
HAVING
критерии_отбора_групп_по_групповым_характеристикам
ORDER BY {столбец|выражение, ... } [ASC|DESC]

Фраза GROUP BY разделяет результаты на подгруппы

Фраза HAVING отбирает часть групп, как правило используя для этого групповые функции

Порядок выполнения фраз:

1. Where
2. Group by
3. Having
4. Order by

Групповые функции

Групповые(агрегатные) функции работают с группой строк и возвращают один результат на группу.

- AVG(DISTINCT|ALL выражение) возвращает среднее значение для группы столбцов.
- MAX (DISTINCT|ALL выражение) возвращает максимум всех значений для группы строк.
- MIN (DISTINCT|ALL выражение) возвращает минимум всех значений для группы строк.
- SUM (DISTINCT|ALL выражение) возвращает сумму всех значений для группы строк
- COUNT ({*|DISTINCT|ALL выражение}) подсчитывает число строк. При задании * функция вычисляет все строки, вне зависимости, имеют ли они конкретное значение или NULL
- STDDEV (DISTINCT|ALL выражение) (Oracle), STDEV(SQL Server) возвращает математическое ожидание в группе.
- VARIANCE (DISTINCT|ALL выражение) возвращает дисперсию в группе.

Данные групповые функции, за исключением COUNT(*) игнорируют NULL-значения.

Фраза DISTINCT заставляет учитывать каждое уникальное значение только один раз.

Примеры использования фразы GROUP BY и групповых функций

```
SELECT department_id, AVG(salary)
FROM employees
GROUP BY department_id;
```

Пример группировки по нескольким столбцам:

```
SELECT department_id dept_id, job_id, SUM(salary)
FROM employees
GROUP BY department_id, job_id;
```

Пример использования фразы HAVING:

```
SELECT department_id, MAX(salary)
FROM employees
GROUP BY department_id
HAVING MAX(salary)>10000;
```

Пример использования вложенных групповых функций:

```
SELECT MAX(AVG(salary))
FROM employees
GROUP BY department_id;
```

Задания

1. Отобразите минимальную, максимальную, суммарную и среднюю зарплату для каждого типа работы. Назовите столбцы Maximum, Minimum, Sum и Averzage соответственно.
2. Определите число менеджеров. Назовите столбец Number of Managers. Подсказка: воспользуйтесь столбцом MANAGER_ID.
3. Отобразите номер менеджера и зарплату наиболее низкооплачиваемого подчиненного данного менеджера. Исключите каждого, чей менеджер неизвестен. Исключите все группы, где минимальная зарплата 6000$ или меньше. Отсортируйте результат в убывающем порядке.

Подзапросы

Подзапрос - это команда SELECT, вложенная в другую команду SELECT для получения промежуточных результатов. Подзапрос выполняется первым и выдает одну или несколько строк

```
SELECT .....
    FROM   табл1
    WHERE сравнение(SELECT   столб2
                    FROM  табл2
                    WHERE  условие )
```

Подзапросы могут находиться во фразах WHERE, HAVING, FROM, ORDER BY команды SELECT. Также в командах INSERT, UPDATE, DELETE

Однострочные подзапросы

Однострочный подзапрос возвращает одну строку.
С однострочными подзапросами используются однострочные операторы сравнения: **>**, **=**, **>=**, **<**, **<>**, **<=**

Пример однострочного подзапроса:

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id=
      (SELECT job_id
      FROM employees
      WHERE employee_id=141)
AND salary>
      (SELECT salary
      FROM employees
      WHERE employee_id=143);
```

Многострочные подзапросы

Многострочный подзапрос возвращает несколько строк

Операторы сравнения для многострочных подзапросов:

IN(подзапрос) - равенство любому из значений

ANY(SOME) - сравнение верно хоть для какого-нибудь значения

ALL – сравнение верно для всех значений

EXISTS – значение существует в подзапросе

NOT EXISTS – значение не существует в подзапросе

Пример использования многострочного подзапроса с оператором сравнения IN:

```
SELECT last_name, salary, department_id
FROM employees
WHERE salary IN (SELECT MIN(salary)
         FROM employees
         GROUP BY department_id);
```

Многострочные подзапросы

Пример использования многострочного подзапроса с оператором сравнения ANY:

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary<ANY (SELECT salary
                  FROM employees
                  WHERE job_id='IT_PROG')
AND job_id<>'IT_PROG';
```

<ANY(меньше хоть одного из значений) эквивалентно <максимального значения.

Пример использования многострочного подзапроса с оператором сравнения ALL:

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary<ALL (SELECT salary
                  FROM employees
                 WHERE job_id='IT_PROG')
AND job_id<> 'IT_PROG';
```

<ALL (меньше всех значений) эквивалентно <минимального значения.

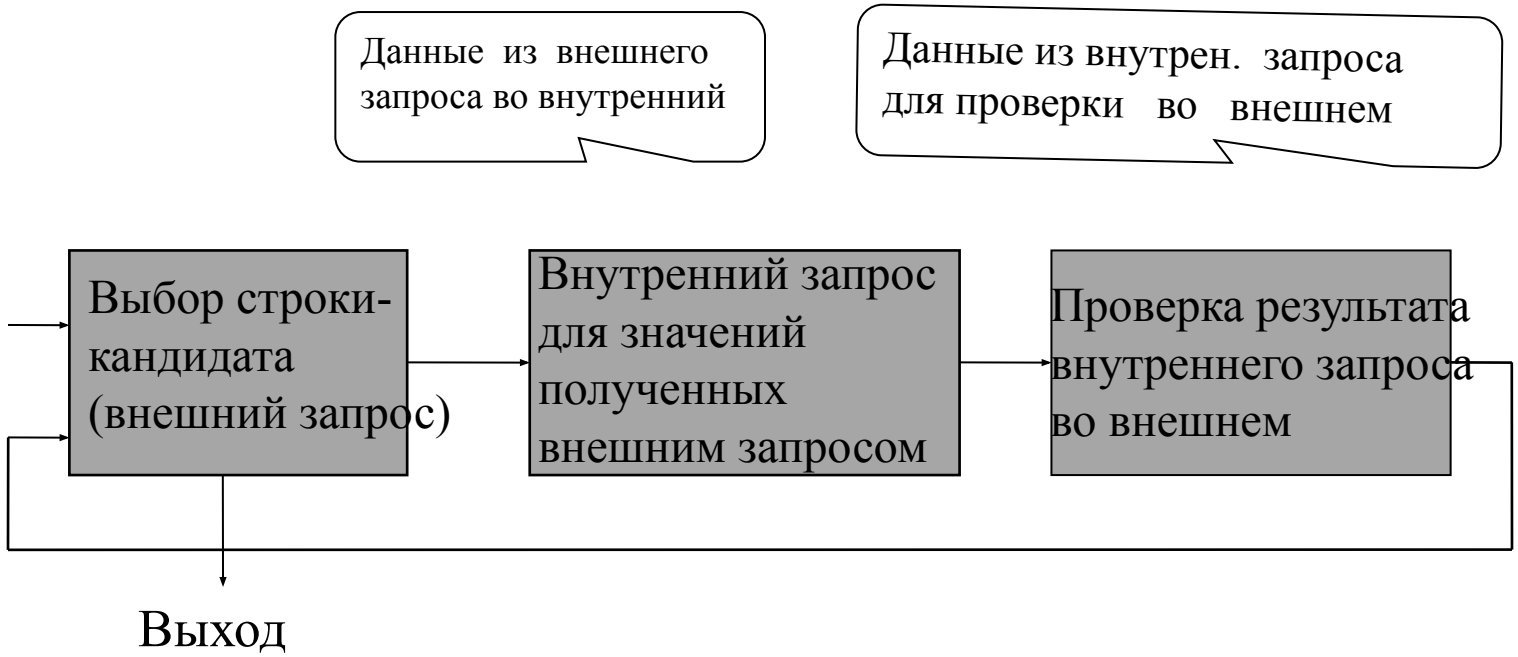
Коррелированные подзапросы

Обычный подзапрос выполняется первым, внешний запрос вторым.

Коррелирующими называются подзапросы, выполняющиеся *для каждой строки-кандидата из внешнего запроса.*

Отсюда вытекает необходимый признак: *Коррелирующий подзапрос содержит столбец из внешнего запроса.*

Процесс выполнения коррелированного запроса:



Пример коррелированного подзапроса

Найти всех работников, которые получают зарплату выше средней в своем отделе:

```
SELECT last_name, salary, department_id
FROM employees outer
WHERE salary>(SELECT AVG(SALARY)
      FROM employees
      WHERE department_id=
      outer.department_id);
```

Использование оператора EXISTS

Оператор EXISTS проверяет, найдена ли хотя бы одна строка. Если да, возвращается TRUE, если нет, то FALSE.

Пример использования оператора EXISTS:

Найти сотрудников, которым подчиняется хотя бы один человек.

```
SELECT employee_id, last_name, job_id, department_id
FROM employees outer
WHERE EXISTS (SELECT 'X'
      FROM employees
      WHERE manager_id=outer.employee_id);
```

Пример использования оператора NOT EXISTS:

```
SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS (SELECT 'X'
           FROM employees
              WHERE department_id=d.department_id);
```

Задания

1. Напишите запрос, отображающий last_name и hire_date всех работников, работающих в том же отделе, что и Zlotkey, исключая Zlotkey.
2. Выведите на экран last_name, department_id и job_ID всех сотрудников, у которых location_id отдела равняется 1700.
3. Напишите запрос, отображающий last_name, department_id и salary каждого работника чей department_id и salary оба совпадают с department_id и salary какого-нибудь работника, получающего комиссионнные.
4. Напишите запрос, выдающий всех сотрудников, которые получают зарплату выше всех менеджеров продаж (JOB_ID='SA_MAN'). Отсортируйте результаты по зарплате от большей к меньшей.
5. Напишите запрос, отображающий last_name тех сотрудников, кто имеет сослуживцев в своих отделах с более поздней датой поступления (hire_date), но с большей зарплатой.

Теоретико-множественные операции

Поддержка операций пересечения и разности появилась только в SQL Server 2005, до этого

были лишь UNION и UNION ALL.

- UNION объединение запросов, выбираются все неповторяющиеся строки

```
SELECT employee_id, job_id
FROM employees
UNION
SELECT employee_id, job_id
FROM job_history;
```

- UNION ALL объединение запросов, выбираются все строки, включая повторяющиеся
- INTERSECT выбираются строки из пересечения
- MINUS в Oracle разность результатов первого и второго запросов

Ограничения целостности

Ограничения целостности это условия специального вида, которые должны выполняться для всей схемы или некоторой **подсхемы** базы данных. Выделяют **декларативные** и **процедурные** ограничения целостности.

Декларативные ограничения описываются заданием некоторого свойства при создании схемы базы. Например, ограничение “первичный ключ” (“primary key”) означает, что значения указанных в определении ключа полей записи определяют ее однозначно.

Процедурные ограничения могут быть определены только через процедуры специального вида, называемые триггерами.

Декларативные ограничения целостности (1/3)

Создаются, когда создается или изменяется таблица.

Определяются на уровне столбца или таблицы.

CREATE TABLE имя_таблицы

(столбец тип_данных [DEFAULT значение] [CONSTRAINT имя_ограничения] тип_ограничения,

…

[CONSTRAINT имя_ограничения] тип_ограничения (столбец, …), …);

Пример создания таблицы с декларативными ограничениями целостности:

```
CREATE TABLE test1(
pk NUMERIC PRIMARY KEY,
fk NUMERIC,
col1 NUMERIC,
col2 NUMERIC,
CONSTRAINT fk_constraint FOREIGN KEY (fk) REFERENCES
  test1,
CONSTRAINT ck1 CHECK (pk>0 and col1>0),
CONSTRAINT ck2 CHECK (col2>0));
```

Декларативные ограничения целостности (2/3)

Добавление декларативного ограничения целостности в существующую таблицу:

ALTER TABLE имя_таблицы
ADD [CONSTRAINT имя_ограничения] тип (столбец);

```
ALTER TABLE TEST1
ADD CONSTRAINT test1_col1_uk UNIQUE(col1);
```

SQL Server есть опция [WITH CHECK|NOCHECK] (после имени таблицы) проверяет на удовлетворение ограничению целостности уже существующих записей.

Удаление ограничения целостности:

Синтаксис Oracle:

ALTER TABLE имя_таблицы
DROP *PRIMARY KEY*| *UNIQUE(столбец)*|CONSTRAINT имя_ограничения *[CASCADE];*

```
ALTER TABLE test1
DROP PRIMARY KEY CASCADE;
```

Декларативные ограничения целостности(3/3)

Отключение ограничений целостности:

Синтаксис Oracle:

ALTER TABLE имя_таблицы
***DISABLE* CONSTRAINT имя_ограничения [CASCADE];**

Синтаксис SQL Server:

ALTER TABLE имя_таблицы *NOCHECK* CONSTRAINT имя_ограничения;

Например:

```
ALTER TABLE test1
NOCHECK CONSTRAINT fk_constraint;
```

Включение ограничений целостности:

Синтаксис Oracle:

ALTER TABLE имя_таблицы
***ENABLE* CONSTRAINT имя_ограничения;**

Задания

1. Создайте таблицу EMP

Имя столбца	ID	NAME	MGR_ID	DEPT_ID
Тип данных	NUMERIC	VARCHAR(20)	NUMERIC	NUMERIC
Ограничение	PRIMARY KEY		FOREIGN KEY, ссылается на ID	

2. Задайте столбцу NAME ограничение NOT NULL
3. Попробуйте вставить строку (1, NULL, 2, 5). Исправьте данные так, чтобы строка вставилась.
4. Добавьте ограничение, проверяющее, что номера отделов кратны 10
5. Отключите ограничение целостности первичного ключа.
6. Включите ограничение целостности первичного ключа.
7. Удалите ограничение целостности CHECK
8. Удалите таблицу EMP

Представления

Представление (View) это виртуальная таблица, сохраняемая в памяти как команда SELECT. Своих данных не содержит и оперирует данными из базовых таблиц. Представления позволяют:

- ограничить пользователю доступ к базе данных, показывая только часть записей и/или не все столбцы;
- упростить формирование запроса пользователем, например, сделав сложное соединение таблиц в виде представления;
- выдавать данные в разных для различных пользователей видах.

Создание представлений (Oracle)

CREATE [OR REPLACE] ***[FORCE]*** **VIEW имя_представления**
[(столбец [, столбец]) .....]
AS
запрос
[WITH CHECK OPTION [CONSTRAINT имя_ограничения]]
[WITH READ ONLY [CONSTRAINT имя_ограничения]];

Опция FORCE позволяет создать представление когда базовые таблицы не существуют или у владельца представления нет к ним доступа

WITH CHECK OPTION ограничивает операции INSERT и UPDATE, выполняемые через представление, чтобы не дать им создать строки, которые само представление не может выбрать

WITH READ ONLY для представления допускаются только выборки

Примеры создания представлений (Oracle)

```
CREATE VIEW salvu50
AS SELECT employee_id ID_NUMBER, last_name NAME,
        salary SALARY
        FROM employees
        WHERE department_id=50;

SELECT * FROM salvu50;

CREATE VIEW dept_sum_vu
(name, minsal, maxsal, avgsal)
AS SELECT d.department_name, MIN(e.salary),
        MAX(e.salary),AVG(e.salary)
    FROM employees e, departments d
    WHERE e.department_id=d.department_id
    GROUP BY d.department_name;
```

Создание представлений(SQL Server)

CREATE VIEW [< имя_базы_данных > .] [< владелец > .]
имя_представления [(столбец [,...n])]
[WITH < view_attribute > [,...n]]
AS
Запрос
[WITH CHECK OPTION]
< view_attribute > ::=
{ ENCRYPTION | SCHEMABINDING | VIEW_METADATA }

ENCRYPTION – шифрование столбца системной таблицы, в котором хранится

текст создания представления

SCHEMABINDING – привязывает представление к схеме

VIEW_METADATA – указывает, что SQL Server вернет метаданные о

представлении, а не о базовых таблицах

В отличии от синтаксиса Oracle отсутствуют фразы FORCE и WITH READ

ONLY, но есть атрибуты ENCRYPTION | SCHEMABINDING |

VIEW_METADATA

Примеры создания представлений SQL Server

```
USE AdventureWorks ;
GO
CREATE VIEW hiredate_view
AS
SELECT c.FirstName, c.LastName, e.EmployeeID,e.HireDate
FROM HumanResources.Employee e JOIN Person.Contact c on
e.ContactID = c.ContactID ;
```

Изменение, удаление представления

Изменение представления:

ORACLE:

- **ALTER VIEW имя_представления COMPILE** если

представление было создано с опцией FORCE

- Также с помощью **ALTER VIEW** можно добавлять, узменять,

удалять ограничения.

Изменение представления с помощью CREATE OR REPLACE

SQL SERVER:

- Синтаксис ALTER VIEW аналогичен CREATE VIEW и позволяет

изменять представление

Удаление представления:

DROP VIEW { *имя_представления* }

В Oracle есть опция cascade constraints

SQL Server позволяет удалять сразу несколько представлений

Обновляемые представления

Можно изменять данные через представление, если оно не содержит:

Oracle:	SQL Server:
•Групповые функции •Ключевое слово DISTINCT •Столбцы, определенные выражениями •Выражение GROUP BY •Псевдостолбец ROWNUM •Столбцы NOT NULL в базовой таблице, которые не выбраны в представлении	•Групповые функции •Ключевое слово DISTINCT не влияет на изменяемые столбцы •Столбцы, определенные выражениям •GROUP BY не влияет на изменяемые столбцы •TOP •UNION, INTERSECT, EXCEPT •FROM ссылается хотя бы на одну таблицу, список столбцов содержит не только не табличные выражения

Изменяться должна только одна таблица из входящих в представление.
Иначе обновление через INSTEAD OF триггеры

Задания

1. Создайте таблицу EMP

Имя столбца	ID	ENAME	SALARY	DEPT_ID
Тип данных	NUMERIC	VARCHAR(20)	NUMERIC	NUMERIC

2. Создайте таблицу DEPT

Имя столбца	DEPT_ID	DNAME
Тип данных	NUMERIC	VARCHAR(20)

3. Создайте представление dept80 над таблицей EMP

4. Вставьте в представление dept80 строку (1, 'Petrov',2000,80)

5. Создайте представление deptName, содержащее все сведения о сотруднике, а также имя его отдела.

6. Попробуйте вставить в deptName строку (2, 'Ivanov', 3000, 30, 'Accounting')

7. Вставьте в представление deptName строку (3, 'Petrov', 4000)

Индексы(Oracle)(1/2)

Индекс – это объект схемы, который может ускорить извлечение строк, используя указатель.
Индексы создаются автоматически, при определении ограничений PRIMARY KEY и UNIQUE или вручную:

CREATE [UNIQUE | BITMAP] INDEX имя_индекса
ON имя_таблицы (столбец1[, столбец2]…);

Фраза **BITMAP** позволяет создавать индексы на основе битовых матриц, которые лучше всего подходят для столбцов с малым числом различных значений.
Фраза UNIQUE создает уникальный индекс, накладывающий ограничение уникальности на каждое значение индекса.

Пример создания индекса:

```
CREATE INDEX emp_last_name_idx
ON  employees(last_name);
```

Индексы(Oracle)(2/2)

Когда создавать индексы:

- Столбец содержит широкий диапазон значений
- Столбец содержит большое число NULL-значений
- Один или несколько столбцов часто используются вместе в условии WHERE или при JOIN
- Таблица большая и большинство запросов возвращают меньше чем 2-4% строк

Удаление: **DROP INDEX имя_индекса;**

Пример удаления индекса:

```
DROP INDEX emp_last_name_idx
```

Индексы SQL Server

Базовый синтаксис создания индекса:

Create [UNIQUE] [CLUSTERED | NONCLUSTERED]

INDEX имя_индекса ON имя_таблицы | имя_представления
(столбец [ASC | DESC] [,…n])

INCLUDE (столбец) [, …n]

UNIQUE создает уникальный индекс. В нем не допускается наличие двух строк с одинаковыми значениями ключа индекса.

CLUSTERED создает индекс, в котором на нижнем уровне хранятся действительные строки данных таблицы. Кластеризованный индекс должен быть уникальным

INCLUDE указывает неключевые столбцы, добавляемые на конечный уровень некластеризованного индекса

Пример создания и удаления индекса в MS SQL сервер.

Создание индекса:

```
USE AdventureWorks;
GO
IF EXISTS (SELECT name FROM sys.indexes WHERE name =
   N'IX_ProductVendor_VendorID') DROP INDEX
   IX_ProductVendor_VendorID ON Purchasing.ProductVendor;
GO
CREATE INDEX IX_ProductVendor_VendorID ON
   Purchasing.ProductVendor (VendorID);
GO
```

Удаление индекса:

```
USE AdventureWorks;
GO
DROP INDEX IX_ProductVendor_VendorID ON
   Purchasing.ProductVendor;
GO
```

Задания

1. Создайте кластеризованный индекс для таблицы EMP на столбец ID
2. Создайте неуникальный индекс для таблицы DEPT на столбец DNAME

Последовательности(Oracle)(1/2)

Генераторы последовательностей позволяют создавать последовательности уникальных значений

CREATE SEQUENCE [пользователь.] имя_последовательности
[INCREMENT BY {1|целое_число|}]
[START WITH целое_число]
[MAXVALUE целое_число |NOMAXVALUE]
[MINVALUE целое_число |NOMINVALUE]
[CYCLE|NOCYCLE]
[CACHE 20|целое_число |NOCACHE]
[ORDER|NOORDER]

Пример создания последовательности:

```
CREATE SEQUENCE dept_deptid_seq
   INCREMENT BY 10
   START WITH 400
   MAXVALUE 9999
   NOCACHE
   NOCYCLE;
```

Последовательности(Oracle)(2/2)

Для генерации очередного значения последовательности используется вызов псевдостолбца **NEXTVAL**, перед которым в качестве префикса всегда стоит имя последовательности
CURRVAL – текущее значение
Пример использования последовательности:

```
INSERT INTO departments (department_id,
  department_name, location_id)
VALUES (dept_deptid_seq.NEXTVAL, 'Support', 2500);
```

Параметры последовательности можно корректировать командой **ALTER SEQUENCE**
Команда удаления последовательности:
DROP SEQUENCE [пользователь.] имя_последовательности

Свойство IDENTITY

В SQL Server нет объекта последовательность. Аналогичную роль играет свойство столбца IDENTITY:
IDENTITY(начальное_значение, инкремент)

Пример использования свойства IDENTITY:

```
USE AdventureWorks
CREATE TABLE new_employees(
id_num int IDENTITY(1,1),
fname varchar (20),
minit char(1),
lname varchar(30));
INSERT new_employees (fname, minit, lname)
VALUES ('Karin', 'F', 'Josephs');
INSERT new_employees (fname, minit, lname)
VALUES ('Pirkko', 'O', 'Koskitalo')
```

Пользователи Oracle

Пользователь характеризуется:

- Имя пользователя
- Пароль
- Привилегии
- Схема
- Профиль(ресурсы)

CREATE USER имя_пользователя
IDENTIFIED BY пароль;

Смена пароля:

ALTER USER имя_пользователя
IDENTIFIED BY пароль;

- Роль – именованная группа привилегий

CREATE ROLE имя_роли;

Примечание. Схема и пользователь у Oracle одно и то же.

Пример создания пользователей

```
CREATE USER vasya
IDENTIFIED BY vasya;
```

Пример изменения пароля:

```
ALTER USER vasya
IDENTIFIED BY qwerty;
```

Системные и объектные привилегии

Существует два вида привилегий: системные и объектные привилегии.

Системные привилегии распространяют разрешение на выполнение различных команд определения данных и управления данными, таких как CREATE TABLE, ALTER TABLE, CREATE USER.

Объектные привилегии распространяют разрешение на действия для определенных именованных объектов базы данных (например, INSERT, UPDATE, DELETE).

Предоставление и изъятие привилегий

Предоставление системных привилегий:

GRANT привилегия [, privilege…]
TO имя_пользователя [, имя_пользователя | роль, PUBLIC…]

Предоставление объектных привилегий:

GRANT объектная_привилегия [(столбцы)]
ON объект
TO {пользователь | роль | PUBLIC}
[WITH GRANT OPTION];

Отмена привилегий

Системных:

REVOKE {привилегия [, привилегия…] | ALL}
FROM {пользователь[, пользователь…] | роль | PUBLIC};

Объектных:

REVOKE {привилегия [, привилегия…] | ALL}
ON объект
FROM {пользователь[, пользователь…] | роль | PUBLIC}
[CASCADE CONSTRAINTS];

Пример предоставления и изъятия привилегий

```
GRANT create session, create table, create sequence,
   create view
TO vasya;
CREATE ROLE programmer;
GRANT create procedure, create trigger
TO programmer;
GRANT programmer
TO vasya;
GRANT select
ON hr.employees
TO vasya;
REVOKE create session
FROM vasya;
```

Пользователи SQL Server

Присоединиться к SQL Server можно только через логин. Если пользователь Windows принадлежит к определенной группе, которая имеет доступ к SQL Server, то он будет иметь доступ к SQL Server. Изначально все права на базу данных принадлежат тому пользователю, который ее создал.

- Создание пользователя Windows
- Создание имени входа

Используется графический интерфейс, можно сгенерировать сценарий

```
USE [master]
GO
CREATE LOGIN [A1\A2] FROM WINDOWS WITH
   DEFAULT_DATABASE=[master]
GO
USE [Sample]
GO
CREATE USER [A1\A2] FOR LOGIN [A1\A2]
GO
USE [Sample]
GO
ALTER USER [A1\A2] WITH DEFAULT_SCHEMA=[dbo]
GO
```

Аналогичные команды можно написать вручную

Фраза Rollup

Расширение к GROUP BY, подсчитывает промежуточные агрегатные значения. Синтаксис фразы в Oracle и SQL Server отличается

SELECT [столбец,] групповая функция (столбец)…
FROM таблица
[WHERE условие]
[GROUP BY [ROLLUP] выражение]
[HAVING условие]
[ORDER BY столбец];
SQL Server:
SELECT [столбец,] групповая функция (столбец)…
FROM таблица
[WHERE условие]
[GROUP BY выражение WITH ROLLUP]
[HAVING условие]
[ORDER BY столбец];

Пример запроса с Rollup

```
select department_id, job_id, sum(salary)
from employees
where department_id<40
group by rollup (department_id, job_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
10		4400
20	MK_MAN	13000
20	MK_REP	6000
20		19000
30	PU_MAN	11000
30	PU_CLERK	13900
30		24900
		48300

1

2

3

Строки, полученные с помощью ROLLUP

1. Строки, совпадающие со строками, полученными с помощью обычной фразы GROUP BY
2. Суммы зарплаты по отделам
3. Сумма всей зарплаты

Фраза Cube

Подсчитывает промежуточные значения для всех возможных комбинаций выражений в предложении group by и общее значение

Синтаксис Oracle:

SELECT [столбец,] групповая функция (столбец)…
FROM таблица
[WHERE условие]
[GROUP BY [CUBE] выражение]
[HAVING условие]
[ORDER BY столбец];

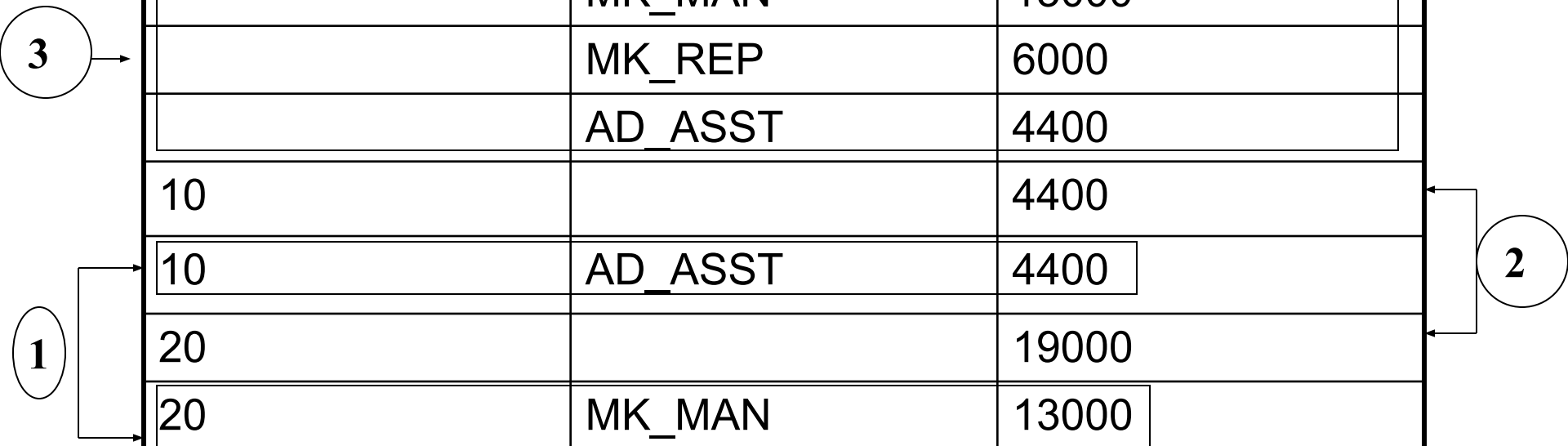
SQL Server:

SELECT [столбец,] групповая функция (столбец)…
FROM таблица
[WHERE условие]
[GROUP BY выражение WITH CUBE]
[HAVING условие]
[ORDER BY столбец];

Пример запроса с Cube

```
select department_id, job_id, sum(salary)
from employees
where department_id<30
group by cube(department_id, job_id);
```

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
		23400
	MK_MAN	13000
	MK_REP	6000
	AD_ASST	4400
10		4400
10	AD_ASST	4400
20		19000
20	MK_MAN	13000
20	MK_REP	6000



Строки, полученные с помощью CUBE

1. Строки, совпадающие со строками, полученными с помощью обычной фразы GROUP BY
2. Агрегатная сумма по отделам
3. Агрегатная сумма по должностям
4. Общая сумма

Работа с иерархиями(Oracle)

SELECT [LEVEL], столбец, выражение…
FROM таблица
[WHERE условия]
[START WITH условия]
[CONNECT BY PRIOR условия];

LEVEL возвращает уровень иерархии

START WITH определяет корень иерархии

CONNECT BY PRIOR определяет отношение между предками и потомками

Условие в WHERE исключает узел

Условие в CONNECT BY исключает ветвь

Пример иерархического запроса

```
SELECT last_name||' reports to
  '||
PRIOR last_name "Walk Top Down"
FROM employees
START WITH last_name='King'
CONNECT BY PRIOR
  employee_id=manager_id;
```

Фраза Merge(Oracle)

Производит UPDATE, если строка существуют или INSERT, если новая строка. Позволяет увеличить производительность, уменьшив число проходов.

```
MERGE INTO имя_таблицы псевдоним
    USING (таблица|представление|подзапрос) псевдоним
    ON (условие соединения)
    WHEN MATCHED THEN
     UPDATE SET
     столбец1=значение1,
     столбец2=значение2
    WHEN NOT MATCHED THEN
    INSERT (список_столбцов)
    VALUES (значения);
```

INTO определяет в какую таблицу вставляем
USING откуда берутся данные для вставки