# Команды SQL

## Правила Э.Ф.Кодда

1. Данные хранятся в столбцах и строках таблиц
2. Доступ к данным можно получить, указав имя таблицы, имя столбца и ключ
3. СУБД должна обрабатывать пропущенные значения (пустые данные). Для обозначения пустых данных используется ключевое слово NULL
4. СУБД должна включать оперативный каталог, содержащий сведения о самой базе данных
5. Для определения данных, их обработки и других операций определяется специальный подъязык. В настоящее время таким языком является SQL
6. Должны поддерживаться представления таблиц, или виртуальные таблицы, которые строятся динамически по запросам
7. Должна быть включена поддержка транзакций. Транзакция обеспечивает коллективное изменение или отмену всех связанных изменений данных. Транзакции имеют четыре свойства: атомарность, согласованность, изолированность и продолжительность (ACID). Атомарность – транзакция либо выполняется полностью, либо не выполняет ничего. Согласованность – с каждой транзакцией БД переходит из одного согласованного состояния в другое согласованное состояние данных. Изолированность – изменения, происходящие в процессе транзакции, не видны пользователю до завершения транзакции. Продолжительность – сделанные в процессе транзакции изменения должны сохраниться в БД
8. Физическое хранение данных отделено от пользователя. Пользователь имеет дело только с логической структурой БД
9. Логическую структуру данных можно изменять с минимальным воздействием на пользователей и программы
10. Правила целостности данных хранятся в каталоге БД. Любые их изменения не должны влиять на прикладные программы
11. Приложения должны работать в распределенной среде (когда данные хранятся в различных местах)
12. СУБД должна обеспечивать безопасность и целостность базы данных

# Команды SQL

- **Команды манипулирования данными (DML)**
- **Команды определения данных (DDL)**
- **Команды управления транзакциями**
- **Команды управления сеансом**
- **Команды управления системой**
- **Встроенные команды**

## 1. Команды манипулирования данными

### 1.1. Команда SELECT

**SELECT [ { ALL | DISTINCT } ]** ***список выбора***
**FROM** ***список таблиц*** **|** ***список представлений***
**[** ***@связь с базой данных*****]**
**WHERE** ***условие***
**START WITH** ***условие***
**CONNECT BY** ***условие***
**GROUP BY** ***поле*** **[,] [ HAVING** ***условие*** **]**
**UNION** ***команда select***
**UNION ALL** ***команда select***
**INTERSECT** ***команда select***
**MINUS** ***команда select***
**FOR UPDATE [ OF** ***поле*** **[,] [ NOWAIT ]]**
**ORDER BY** ***поле*** **|** ***позиция*** **[ ASC | DESC ]**

**Oracle обеспечивает целостность выполнения каждого предложения SELECT. СУБД обрабатывает запрос так, чтобы данные в таблицах-источниках соответствовали моменту старта выполнения SELECT.**

**Логика обработки запроса:**

1) **FROM**
2) **CONNECT BY**
3) **WHERE (здесь вычисляется ROWNUM)**
4) **GROUP BY**
5) **HAVING**
6) **ORDER BY**
7) **SELECT (здесь могут применяться аналитические функции)**

## 1.1.1. Использование псевдостолбцов

| Псевдостолбец | Возвращаемое значение |
|---|---|
| sequence.CURRVAL | Последнее значение последовательности. |
| sequence.NEXTVAL | Следующее значение последовательности. |
| LEVEL | Глубина запроса внутри дерева. |
| ROWID | Точное расположение строки данных в памяти. |
| ROWNUM | Порядковый номер выбранной строки |
| SYSDATE | Текущая дата и время |
| UID | Уникальный идентификатор текущего пользователя |
| USER | Имя, под которым пользователь зарегистрировался в базе данных. |

## 1.1.2. Рекурсивные запросы

```
EMPNO ENAME      JOB           MGR     DEPTNO
----------  ----------    ---------   ----------  ----------
 7369   SMITH       CLERK          7902        20
 7566   JONES       MANAGER     7839        20
 7782   CLARK       MANAGER    7839        10
 7788   SCOTT       ANALYST     7566        20
 7839   KING        PRESIDENT                10
 7876   ADAMS       CLERK          7788        20
 7902   FORD        ANALYST     7566        20
 7934   MILLER      CLERK          7782       10
```

```
SELECT empno, SUBSTR(LPAD(' ',(LEVEL-1)*2)||ename, 1, 15)
name
   FROM emp
   START WITH mgr is null
   CONNECT BY PRIOR empno = mgr;
```

```
  EMPNO NAME
  ----------   ---------------
    7839   KING
    7566      JONES
    7788         SCOTT
    7876            ADAMS
    7902         FORD
    7369            SMITH
    7782      CLARK
    7934          MILLER
```

## 1.1.3. Использование ROLLUP

```
SELECT count(a.amt) cnt_all FROM ord a;
SELECT a.snum,count(a.amt) cnt_all FROM ord a
  GROUP BY a.snum
  ORDER BY a.snum;
SELECT a.snum,a.odate,count(a.amt) cnt_all FROM ord a
  GROUP BY a.snum,a.odate
  ORDER BY a.snum,a.odate;

SELECT a.snum,a.odate,count(a.amt) cnt_all FROM ord a
    GROUP BY ROLLUP(a.snum,a.odate);
```

| SNUM | ODATE | CNT_ALL |
|---|---|---|
| 1001 | "03-OCT-90" | 1 |
| 1001 | "05-OCT-90" | 1 |
| 1001 | "06-OCT-90" | 1 |
| 1001 | | 3 |
| 1002 | "03-OCT-90" | 1 |
| 1002 | "04.01.2010" | 1 |
| 1002 | "06.01.2010" | 1 |
| 1002 | | 3 |
| 1003 | "04-OCT-90" | 1 |
| 1003 | | 1 |
| 1004 | "03-OCT-90" | 1 |
| 1004 | | 1 |
| 1007 | "03-OCT-90" | 2 |
| 1007 | | 2 |
| | | 10 |

## 1.2. Команда INSERT

```
INSERT INTO  [ схема. ] { таблица | представление }
             [ @связь с базой данных] (поле [,])
             { VALUES ( выражение [,] ) | запрос }


      INSERT INTO emp (empno, ename)
            VALUES (7777, 'BILL');

      INSERT INTO emp_copy
            SELECT * FROM emp;
```

## 1.3. Команда UPDATE

**UPDATE [ *схема.* ] { *таблица* | *представление* }**
**[ @*связь с базой данных*] [ *алиас* ]**
**SET { *поле* = { *выражение* | *запрос* } [,] | ( *поле* [,]) = *запрос* }**
**[ WHERE *условие* ]**

```
UPDATE emp  SET sal =sal * 1.1
      WHERE job = 'CLERK';

UPDATE emp_copy ec      SET (ename, job, mgr)
      = (SELECT e.ename, e.job, e.mgr FROM emp e
            WHERE e.empno = ec.empno)
      WHERE ename LIKE 'A%';
```

## 1.4. Команда DELETE

**DELETE [ FROM ] [ *схема.* ] { *таблица* | *представление* }**
**[ @*связь с базой данных*] [ *алиас* ]**
**[ WHERE *условие* ]**

# 2. Команды определения данных

**Неявно подтверждают текущую транзакцию.**
**Не поддерживаются в PL/SQL**

- **таблицы**
- **индексы**
- **кластеры**
- **последовательности**
- **представления**
- **каналы связи базы данных**
- **синонимы**

## 2.1. Таблицы

**CREATE TABLE [ схема. ] *имя_таблицы***
**(*имя_столбца тип_данных* [DEAFAULT *выражение*] [*ограничение_столбца*] [,])**
**CLUSTER *имя_кластера* ( имя_столбца [,] )**
**ENABLE [ { ALL TRIGGERS | (ограничения целостности) }**
**DISABLE [ { ALL TRIGGERS | (ограничения целостности) }**
**TABLESPACE имя_табличного_пространства**
**(распределение экстента)**
**AS запрос**

```
CREATE TABLE mytab (mycol VARCHAR2(1))
        STORAGE (INITIAL  200 K
                NEXT  20 K
                MINEXTENTS  1
                MAXEXTENTS  100
                PCTINCREASE  20
                     PCTFREE  5
                PCTUSED  30);

ALTER TABLE mytab
        STORAGE (NEXT 30 K
                MAXEXTENTS  110
        PCTINCREASE  0);
```

**DROP TABLE [[ схема.] *имя_таблицы***
**[CASCADE CONSTRAINTS]**

# 2.2. Индексы

## Типы индексов

- **уникальные**
- **неуникальные**

**CREATE INDEX [ схема. ] имя_индекса ON**
**[ схема. ] имя_таблицы (имя_столбца [,] [{ ASC | DESC }])**
**CLUSTER [ схема. ] имя_кластера**
**NOSORT**
**TABLESPACE имя_табличного_пространства**
**(распределение памти)**

```
CREATE INDEX emp_ind ON emp(ename)
        STORAGE (INITIAL 20 K
                 NEXT 20 K
                 PCTINCREASE 0);
```

**ALTER INDEX [схема.] имя_индекса**
**(распределение памяти)**

**DROP INDEX [схема.] имя_индекса**

**Индекс, который должен существовать в любой таблице - индекс, созданный посредством ограничения PRIMARY KEY или UNIQUE**

**Оптимизаторы Oracle - специальные средства, которые решают, как лучше выполнять доступ к данным**

## 2.3. Кластеры

- **индексные кластеры**
- **хешированные кластеры**

**CREATE CLUSTER [schema.]cluster**
**(*распределение памяти*)**
**[SIZE integer [K|M] ]**
**[INDEX | [HASH IS column] HASHKEYS integer]**

```
CREATE CLUSTER emp_dep_clu
        (d_no NUMBER(2));
CREATE INDEX emp_del_ind ON CLUSTER emp_dep_clu;
CREATE TABLE emp
        AS SELECT * FROM scott.emp
        CLUSTER emp_dep_clu(depno);
CREATE TABLE dept
        AS SELECT * FROM scott.dept
        CLUSTER emp_dep_clu(depno);


CREATE CLUSTER emp_hash
        (empno NUMBER(4))
        SIZE 500
        HASH IS empno
        HASHKEYS 300;
CREATE TABLE emp
        AS SELECT * FROM scott.emp
        CLUSTER emp_hash(empno);
```

**ALTER CLUSTER [*схема*.] *имя_кластера***
**(*распределение памяти*)**

**DROP CLUSTER [*схема*.] *имя_кластера***
**[INCLUDING TABLES [CASCADE CONSTRAINTS] ]**

## 2.4. Представления

**CREATE [ OR REPLACE ] VIEW [*схема*.] *имя_представления***
**AS select-*команда***
**[ WITH READ ONLY | WITH CHECK OPTION ]**

**ALTER VIEW [*схема*.] *имя_представления*** **COMPILE**

**DROP VIEW [*схема*.] .] *имя_представления***

## 2.5. Последовательности

**CREATE SEQUENCE [ *схема.* ] *имя_последовательности***
**[ INCREMENT BY integer ]**
**[ START WITH integer ]**
**[ MAXVALUE integer | NOMAXVALUE ]**
**[ MINVALUE integer | NOMINVALUE ]**
**[ CYCLE | NOCYCLE ]**
**[ CACHE integer | NOCACHE ]**
**[ ORDER | NOORDER ]**

**ALTER SEQUENCE [*схема.*] *имя_последовательности***
**[INCREMENT BY integer]**
**[MAXVALUE integer | NOMAXVALUE]**
**[MINVALUE integer | NOMINVALUE]**
**[CYCLE | NOCYCLE]**
**[CACHE integer | NOCACHE]**
**[ORDER | NOORDER]**

**DROP SEQUENCE [*схема.*] *имя_последовательности***

## 2.6. Каналы связи базы данных

**CREATE [PUBLIC] DATABASE LINK *имя_канала_связи***
**[CONNECT TO *пользователь* IDENTIFIED BY *пароль*]**
**[USING 'спецификация_удаленной_базы_данных']**

**SELECT * FROM emp@mylink;**

**DROP [PUBLIC] DATABASE LINK *имя_канала_связи***

## 2.7. Синонимы

**CREATE [ PUBLIC ] SYNONYM [*схема.*] *имя_синонима***
**FOR [ *схема.* ] *имя_объекта***

**DROP [PUBLIC] SYNONYM [*схема.*] *имя_синонима***

# 3. Команды управления транзакциями

- **COMMIT**
- **ROLLBACK**
- **SAVEPOINT**
- **SET TRANSACTION**

## 3.1. Понятие транзакции

**ТРАНЗАКЦИЯ - это логическая единица работы, составленная из одной или нескольких команд SQL**

***Банковская транзакция***

**Транзакция начинается**
**Уменьшить накопительный счет**
**UPDATE savings_accounts SET balance = balance - 500**
**WHERE account = 3209;**
**----------**
**Увеличить чековый счет**
**UPDATE checking_accounts SET balance = balance + 500**
**WHERE account = 3208;**
**----------**
**Записать в журнале банковских операций**
**INSERT INTO journal**
**VALUES(journal_seq.NEXTVAL,'1B',3209, 3208, 500);**
**----------**
**Закончить транзакцию**
**COMMIT WORK;**
**----------**
**Транзакция заканчивается**

## 3.2. ORACLE и управление транзакциями

**Транзакция в ORACLE начинается, когда встречается первая выполнимая команда SQL.**

**Транзакция заканчивается, когда происходит одно из следующих событий:**

- **выдана команда COMMIT**
- **выдана команда ROLLBACK (без фразы SAVEPOINT)**
- **выдана команда DDL (CREATE, DROP, RENAME, ALTER, …).**
- **пользователь отсоединяется от ORACLE. (транзакция подтверждается.)**
- **имеет место аварийное прекращение пользовательского процесса. (транзакция откатывается.)**

**После окончания одной транзакции очередная выполнимая команда SQL автоматически начинает следующую транзакцию.**

## 3.3. Подтверждение транзакций

**COMMIT WORK;**
**COMMIT;**

## 3.4. Откат транзакций

**ROLLBACK WORK;**
**ROLLBACK;**

**ROLLBACK TO SAVEPOINT *имя_точки_сохранения*;**
**ROLLBACK TO *имя_точки_сохранения*;**

## 3.5. Точки сохранения

**SAVEPOINT *имя_точки_сохранения*;**

| Предложение SQL | Результаты |
|---|---|
| SAVEPOINT a; | Первая точка сохранения в транзакции. |
| DELETE ... ; | Первое предложение DML в транзакции. |
| SAVEPOINT b; | Вторая точка сохранения в транзакции. |
| INSERT INTO ; | Второе предложение DML в транзакции. |
| SAVEPOINT c; | Третья точка сохранения в транзакции. |
| UPDATE ... ; | Третье предложение DML в транзакции. |
| ROLLBACK TO c; | Предложение UPDATE откатывается, точка сохранения C остается определенной. |
| ROLLBACK TO b; | Предложение INSERT откатывается, точка сохранения C теряется, точка сохранения B остается определенной. |
| ROLLBACK TO c; | Ошибка - точка сохранения C больше не определена. |
| INSERT INTO ... ; | Новое предложение DML в транзакции. |
| COMMIT; | Подтверждает все действия, выполненные первым предложением DML в транзакции (DELETE) и последним (вторым предложением INSERT).Все прочие предложения (второе и третье предложения DML) в транзакции были подвергнуты откату перед этим COMMIT. |

## 3.6. Типы транзакций

### 3.6.1. Транзакции чтения/записи

**SET TRANSACTION READ WRITE**

### 3.6.2. Транзакции только чтения

**SET TRANSACTION READ ONLY**

### 3.6.3. Дискретные транзакции

```
EXECUTE dbms_transaction.begin_discrete_transaction;
INSERT …;
INSERT …;
COMMIT;
```

## 3.7. PL/SQL и транзакции

```
BEGIN
DELETE FROM tab_1;
DELETE FROM tab_2;
RAISE_APPLICATION_ERRORE(-20001,'Forced Errore');
END
/
```

## 3.8. Локальные, удаленные и распределенные транзакции

```
UPDATE emp  SET job = 'ANALYST'
            WHERE emono = 7934;
COMMIT;


UPDATE dept@kadry.dbs  SET location = 'LONDON'
WHERE deptno = 20;
COMMIT;


UPDATE emp  SET job = 'ANALYST'
WHERE emono = 7934;
UPDATE dept@kadry.dbs  SET location = 'LONDON'
WHERE deptno = 20;
COMMIT
```

## 3.9. Автономные транзакции

```
CREATE OR REPLACE PACKAGE BODY lib  IS
   . . .
   PROCEDURE saveline (code IN INTEGER, text IN VARCHAR2)  IS
        PRAGMA AUTONOMOUS_TRANSACTION;
   BEGIN
        INSERT INTO tab_log(cd, txt) VALUES(code, text);
        COMMIT;
        EXCEPTION WHEN OTHERS THEN
        ROLLBACK;
   END;
   . . .
END;

BEGIN
   .  . .
EXCEPTION WHEN OTHERS THEN
   lib.saveline (SQLCODE, SQLERRM)  ;
END;


CREATE OR REPLACE TRIGGER bef_ins_ord
   BEFORE INSERT ON ord
   FOR EACH ROW
   DECLARE
   PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
   INSERT INTO tab_log
   VALUES(:new.onum,:new.amt,'INSERT',USER,SYSDATE);
   COMMIT;
END;
```

# 4. Команды управления сеансом

**Неявно не подтверждают текущую транзакцию.**
**Не поддерживаются в PL/SQL**

- **ALTER SESSION**
- **SET ROLE**

```
ALTER SESSION
   { SET
     { SQL_TRACE                    = { TRUE | FALSE }
       | GLOBAL_NAMES                 = { TRUE | FALSE }
       | NLS_LANGUAGE                 =   language
       | NLS_TERRITORY                =   territory
       | NLS_DATE_FORMAT              =  'fmt'
       | NLS_DATE_LANGUAGE            =   language
       | NLS_NUMERIC_CHARACTERS =  'text'
       | NLS_ISO_CURRENCY             =   territory
       | NLS_CURRENCY                 =  'text'
       | NLS_SORT                     = { sort  | BINARY }
       | LABEL              = {'text' | DBHIGH | DBLOW | OSLABEL }
       | MLS_LABEL_FORMAT             =  'fmt'
       | OPTIMIZER_GOAL = { RULE | ALL_ROWS | FIRST_ROWS | CHOOSE }
       | FLAGGER = { ENTRY | INTERMEDIATE | FULL | OFF }
       | CLOSE_CACHED_OPEN_CURSORS = { TRUE | FALSE }
     } ...
       | CLOSE DATABASE LINK dblink
       | ADVISE {COMMIT | ROLLBACK | NOTHING}
       | {ENABLE | DISABLE} COMMIT IN PROCEDURE
   }
```

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'DD.MM.YYYY HH24:MI:SS'
```

```
SET ROLE     { role [IDENTIFIED BY password]
          [, role [IDENTIFIED BY password] ] ...
           | ALL [EXCEPT role [, role] ...]
           | NONE }
```

# 5. Команды управления системой

**Неявно подтверждают текущую транзакцию.**
**Не поддерживаются в PL/SQL**

```
ALTER SYSTEM
   { {ENABLE | DISABLE} RESTRICTED SESSION
   | FLUSH SHARED_POOL
   | {CHECKPOINT | CHECK DATAFILES} [GLOBAL | LOCAL]
   | SET { RESOURCE_LIMIT          = { TRUE | FALSE }
       | GLOBAL_NAMES            = { TRUE | FALSE }
       | MTS_DISPATCHERS         = 'protocol, integer'
       | MTS_SERVERS             = integer
       | LICENSE_MAX_SESSIONS    = integer
       | LICENSE_SESSIONS_WARNING = integer
       | LICENSE_MAX_USERS       = integer
       | SESSION_CACHED_CURSORS  = integer } ...
   | SWITCH LOGFILE
   | {ENABLE | DISABLE} DISTRIBUTED RECOVERY
   | ARCHIVE LOG archive_log_clause
   | KILL SESSION 'integer1, integer2' }
```

# 6. Встроенные команды

**Команды встроенного SQL позволяют помещать предложения языка управления данными (DDL), языка манипулирования данными (DML) и управления транзакциями в программу на процедурном языке. Встроенный SQL поддерживается прекомпиляторами ORACLE.**