Езиците в СУБД

Езикът SQL (Structured Query Language)

SQL-92 е разработен на базата на SEQUEL (IBM) и в момента има два публикувани стандарта:

- ANSI X3.135-1992, “Database Language SQL”
- ISO/IEC 9075:1992, “Database Language SQL”

В тях са дефинирани 4 нива на езика: "Entry”, “Transitional”, “Intermediate” и “Full”. Всяка реализация на SQL трябва да поддържа най-малко ниво "Entry”.

За да илюстрираме свойствата на езика ще използваме следните релации:

```
CREATE TABLE dept
(deptno NUMBER(2) CONSTRAINT pk_dept PRIMARY KEY,
dname VARCHAR2(14),
loc VARCHAR2(13) );
CREATE TABLE emp
(empno NUMBER(4) CONSTRAINT pk_emp PRIMARY KEY,
ename VARCHAR2(10),
job VARCHAR2(9),
mgr NUMBER(4),
hiredate DATE,
sal NUMBER(7,2),
comm NUMBER(7,2),
deptno NUMBER(2) CONSTRAINT fk_deptno REFERENCES dept );
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500		30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

Типове данни

Символни типове

Char за отделни символи.

Char(n) за низове от *n* символа (**Varchar** при DB2)

Varchar (при ORACLE) за низове за връзка с процедурните езици.

Long (ORACLE) за низове от максимум 65655 символа.

В ACCESS те са Text, Memo.

Числени типове

number, number(n) (**float** при SQL/DS) за цели с променлива или фиксирана дължина. **number(m,n)** (**Decimal** при SQL/DS) за реални числа с дължина *m* и с *n* цифри след десетичната точка. В ACCESS те са **Number**(**Integer**, **LongInteger**, **Byte**, **Single**, **Double**) , **Currency**, **Autonumber**

Други типове

Date (**Time** в SQL/DS) представя дата като сложна структура от полета. **Raw** (ORACLE V6) за двоични данни, между които и bitmaps. **BLOB** (Binary Large OBjects). В ACCESS те са **Date/Time**, **Logical**, **OLE** обекти.

Предикати, функции и изрази

Операции

– Сравнения чрез = **!= > >= < <=**

– BETWEEN / NOT BETWEEN за интервал от стойности

```
SELECT ename FROM emp WHERE hiredate BETWEEN 1.1.80 AND 31.12.80; /* Query1 */
```

– IN / NOT IN за принадлежност на множество (списък) от стойности.

```
SELECT ename FROM emp WHERE job In ('CLERK','MANAGER'); /* Query2 */
```

– 1S NULL / IS NOT NULL за проверка на неопределена стойност

```
SELECT ename FROM emp WHERE comm IS NOT NULL; /* Query3 */
```

– EXISTS / NOT EXISTS за проверка за съществуване на поне един кортеж

– LIKE/NOT LIKE за сравнение на символни низове
  % замества 0 или повече символи, _ замества точно един символ.
  В ACCESS съответните символи са * и ?.

пример: LIKE 'TARKO%', LIKE '%WSKI', LIKE 'A_C'

```
SELECT ename, job FROM Emp WHERE.ename LIKE "b*";/* Query4 */
```

– ANY – кой да е

– SOME – някой

```
SELECT ename FROM emp WHERE sal >ANY (SELECT sal FROM emp WHERE deptno = 20);
/* Query5*/
```

Връща имената на служителите, които имат заплата по-голяма от кой да е от служителите от отдел с код 20.

– ALL – всички

```
SELECT ename FROM emp WHERE sal > ALL (SELECT sal FROM emp WHERE deptno = 20);
/* Query6 */
```

Връща имената на служителите, които имат заплата по-голяма от всички служители от отдел с код 20.

– NOT, AND, OR – са логически операции

```
SELECT ename FROM Emp WHERE job =’CLERK' AND sal>500; /* Query7 */
```

– UNION
  Обединение на 2 заявки, връщащ резултат със същата: SELECT .... UNION SELECT ....

```
SELECT ename FROM query2 UNION ALL SELECT ename FROM query7 /* Query8 */
```

- INTERSECT
  Сечение на 2 заявки с една и съща схема : SELECT .... INTERSECTION SELECT ...
- MINUS
  Разлика на 2 заявки с една и съща схема : SELECT .... MINUS SELECT ...
- DISTINCT – Елиминира дублираните кортежи от резултата:

SELECT DISTINCT job FROM emp; /* Query9 */

Код	**Операция**
+, -	Положително, обратен знак
*, /	Умножение, деление
+, -, || (&)	Събиране, изваждане, конкатенация
=, !=, <, >, <=, >=, IS NULL, LIKE, BETWEEN, IN	сравнения
^, NOT	Степен, отрицание
AND	конюнкция
OR	дизюнкция

Числови функции

- ABS(n) – Абсолютна стойност; CEIL(n) – най-малкото цяло, по-голямо от аргумента; FLOOR(n) – най-голямото цяло, по-малко от аргумента.
- ROUND(n[,m]) – закръгляване с *m* цифри след точката.
- TRUNC(n[,m]) – отрязване с *m* цифри след точката.
- MOD(n) – модул (остатък)
- POWER(m, n) – m^n, с *n* цяло.
- SQRT(n)

Функции за обработка на символи.

- ASCII(char) връща ASCII кода на символа (ASC в ACCESS)
- CHR(n) връща символа, чийто ASCII код е параметър.
- INITCHAR(string) връща низа с като прави първата буква главна.(липсва в ACCESS)
- LENGTH(string) броя на символите в низ.
- LPAD(string, n, char), RPAD(string, n, char) запълва наляво или надясно низа с *n* екземпляра на символа *char.* (SPACES(n) връща *n* интервала в ACCESS)
- LTRlM(string, char), RTRIM(string, char) премахва всичко, което се намира наляво или надясно от символа char, ако той се намира в низа.
- TRANSLATE(string, cl, c2) замества в низа всички появявания на символа cl с c2 (липсва в ACCESS).
- SUBSTR(string, pos, length) връща подниз със зададена дължина започващ от pos. MID$(string, pos, length) в ACCESS)
- UPPER(string), LOWER(string) конвертира низа с главни или малки букви.(UCASE, LCASE в ACCESS)
- || е операция за конкатенация на низове.(В ACCESS (VB) операторът е &)

Агрегатни функции.

Те позволяват да се пресмятат обобщения за множества от кортежи

- COUNT(*) връща броя на кортежите; COUNT(attribute) : връща броя на стойностите на атрибута (без NULL стойностите).
- SUM(attribut) връща сумата на стойностите на атрибута.
- AVG(attribut) връща средната стойност на стойностите на атрибута.
- MIN(attribut), MAX(attribut) връщат минималната и максималната на стойност на атрибута

– VARIANCE(attribut) връща вариацията на стойностите на атрибута.(VAR en ACCESS)

SELECT count(*) FROM emp WHERE deptno=20; /* Query10 */

SELECT AVG(sal) FROM emp WHERE deptno=20; /* Query11 */

Функции на конвертиране на типовете

– TO_CHAR(n [,format]) обръща число *n* в символен низ според зададения формат(STR$ и FORMAT в ACCESS)
– TO_CHAR(date[, format]) обръща дата в низ (STR в ACCESS)
– TO_DATE(string,format) обръща низ в дата (CDATE в ACCESS)
– TO_NUMBER(string) обръща низ в цяло или реално число.(VAL в ACCESS)
Примери за формати : 'DD-MON-YY', 'DD/MM/YYYY', 'MONTH **DD,YYYY'**

В Interbase функцията за конверсия е CAST(*value* **AS** *datatype***)**

Други функции

DECODE(expression, v1,r1[,v2,r2[,v3,r3]]) връща r1,ако изразът expression има стойност, в противен случай връща r2, ако изразът е v2 и.т.н.

NVL(expression1, expression2) връща expression2, ако expression1 е NULL (NZ в ACCESS)

GREATEST(e1, e2 ...), LEAST(e1, e2,...) връщат екстремалните стойности от списък.

В Interbase се поддържат само функциите COUNT, SUM, CAST, AVG, UPPER, MAX, MIN

Език за манипулиране на данните (DML).

Частта DML от езика, която се отнася до извличането на информация, се осигурява от оператора SELECT, чийто синтаксис е:

SELECT [DISTINCT | ALL]{* | израз| списък от атрибути } FROM <table [alias]>,... [WHERE условие за селекция или съединение] [CONNECT BY колона [START WITH условие] [GROUP BY списък от атрибути] [HAVING условие за селекция]

[UNION | INTERSECT | MINUS SELECT...] [ORDER BY списък от атрибути [ASC | DESC]];

Проекция.

Това нормално е извличането на определени колони с елиминирането на дублираните кортежи, но SQL не ги отстранява, освен ако не поискано явно с ключовата дума DISTINCT.

SELECT job, mgr FROM emp; /* Query12 */

SELECT DISTINCT job, mgr FROM emp; /* Query13 */

Избор

Това е извличане на редове според критерий, който е изразен в клаузата WHERE на SELECT:

SELECT * FROM emp WHERE deptno=10; /* Query14 */

Селекция

Това е избор, следван от проекция.

```
SELECT ename, job, sal  FROM emp WHERE job = 'MANAGER' AND
sal>1000; /* Query15 */
SELECT ename, job,sal FROM emp WHERE ename LIKE '%mi%'; /*Query16*/
```

Декартово произведение

SELECT • FROM emp, dept; /* Query17 */

Съединение с квалификации

SELECT * FROM emp,dept WHERE emp.deptno = dept.deptno; /* Query18 */

или на ACCESS или Interbase:

SELECT * FROM emp INNER JOIN dept ON emp.deptno = dept.deptno; /* Query19 */

Тази операция е еквисъединение.

On peut renommer le nom d'une table (ou d'une colonne) dans une requête à l'aide d'un alias (pseudonyme) plus simple à manipuler. Spécialement pratique avec les jointures.

Може да се дадат псевдоними на таблиците в една заявка, които да са по-прости и кратки. Това често се практикува в съединенията:

```
SELECT * FROM emp E,dept D WHERE E.deptno = D.deptno; /*Query20 */
```

или на ACCESS или Interbase:

```
SELECT * FROM emp as E INNER JOIN dept as D ON E.deptno = D.deptno;
/* Query21 */
```

Ограничение върху съединение

```
SELECT ename, job, dept.deptno, dname FROM emp, dept WHERE
emp.deptno = dept.deptno AND job = 'CLERK'; /* Query22 */
```

или на ACCESS или Interbase:

```
SELECT ename, job, dept.deptno, dname FROM emp INNER JOIN dept ON
emp.deptno = dept.deptno WHERE  job = 'CLERK';
```

Тази операция е θ -съединение

Подзаявки

SELECT ename FROM emp WHERE deptno IN (SELECT deptno FROM dept WHERE dname LIKE '%S%') ; /* Query23 */

позволява да се търсят служителите от отделите, чието име съдържа "S". Операцията IN търси ключовете от резултантните кортежи на подзаявката. Око операцията е различна от равенство се използва оператор за сравнение:

SELECT ename, sal FROM emp WHERE deptno = 10 AND sal > (SELECT MAX(sal) FROM emp WHERE deptno = 20); /* Query24 */

SELECT dname, deptno FROM dept WHERE EXISTS (SELECT * FROM emp WHERE dept.deptno = emp.deptno); /*Query 35*/

Корелативни подзаявки

Основната заявка дава една след друга стойностите за подзаявката (подзаявката се изпълнява за всеки кортеж от главната заявка).

Служителите, чиито заплати са по-големи от средната за техния отдел:

SELECT ename, deptno FROM emp E1 WHERE E1.sal>(SELECT AVG(E2.sal) FROM emp E2 WHERE E1.deptno=E2.deptno); /* Query25 */

Всички отдели, в които има служители, имащи заплати по-високи от средната на отдел "ACCOUNTING".

SELECT dname, deptno FROM dept AS D2 WHERE EXISTS (SELECT * FROM emp E, dept D WHERE E.deptno=D.Deptno and D2.deptno = D.deptno and E.sal > (SELECT AVG(sal) FROM emp E1, dept D1 WHERE D1.dname LIKE 'ACCOUNTING')) /* Query32 */;

Групиране

GROUP BY позволява да се групират резултатите според даден критерий и кортежите от всяка група да се обработват от агрегатни функции. Тази клауза се прилага за тези атрибути, които не са параметри на агрегатна функция!!

```
SELECT deptno, MIN(sal), MAX (sal) FROM emp GROUP BY deptno; /*
Query26 */
SELECT deptno, MIN(sal), MAX (sal)FROM emp WHERE job = 'CLERK'
GROUP BY deptno; /* Query27 */
```

HAVING позволява да се изразят критерии спрямо групите (агрегатните операции) и се използва само с GROUP BY

```
SELECT deptno, MIN(sal), MAX(sal) FROM emp GROUP BY deptno HAVING
MAX(sal) > 1200 ; /* Query28 */
SELECT deptno, MIN(sal), MAX(sal) FROM emp WHERE job = 'CLERK'
GROUP BY deptno HAVING MAX(sal) < 1200 ; /* Query29 */
```

Редът на изпълнение е следния: :

1. Ако има клауза WHERE СУБД премахва всички кортежи, които не удовлетворяват условието.
2. Извършва групирането и изчислява агрегатните стойности.
3. Премахва всички групи, които не удовлетворяват условието в клаузата HAVING.

По-сложен пример – да се намерят процентните отношения на броя на служителите и на сумата на техните заплати за всеки отдел спрямо общите за цялото предприятие:

В ORACLE :

```
SELECT a.deptno ”Department”, a.num_emp/b.total_count ”%Employees”,
a.sal_sum/b.total_sal ”%Salary”
FROM (SELECT deptno, COUNT(*) num_emp, SUM(SAL) sal_sum FROM emp
GROUP BY deptno) a,
(SELECT COUNT(*) total_count, SUM(sal) total_sal FROM emp) b ;
```

В SQL ACCESS и Interbase :

```
CREATE VIEW X AS SELECT deptno, COUNT(*) num_emp, SUM(SAL) sal_sum
FROM emp
GROUP BY deptno; /*Query35*/
```

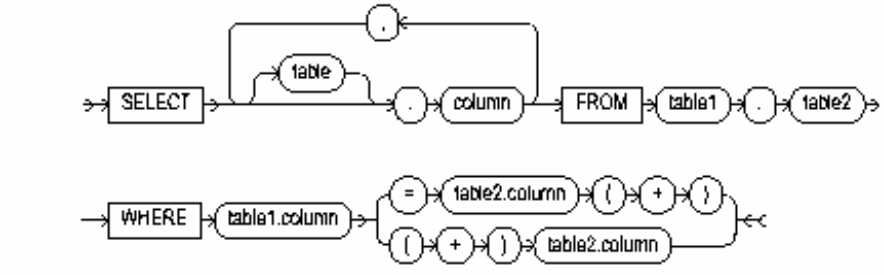
CREATE VIEW Y AS `SELECT COUNT(*) total_count, SUM(sal) total_sal FROM emp); /*Query36*/`

```
SELECT x.deptno AS Department, x.num_emp/y.total_count AS
Pr_Employees, x.sal_sum/y.total_sal AS Pr_Salary
FROM X, Y; /* Query37 */
```

Външно съединение

Този оператор е стандартизиран в по-новите версии на SQL и затова е реализиран по различен начин в различните СУБД.

В Oracle синтаксисът е:



В ACCESS и Interbase :

```
FROM table …]{LEFT | RIGHT | FULL } [OUTER]} JOIN table ON comparaison
```

Пример :

```
SELECT ename, job, dept.deptno, dname
FROM emp, dept
WHERE emp.deptno (+) = dept.deptno;
В ACCESS и Interbase
SELECT emp.ENAME, emp.JOB, dept.DEPTNO, dept.DNAME
FROM emp RIGHT JOIN dept ON emp.DEPTNO = dept.DEPTNO;
```

По-смислен пример – броят на служителите във всеки отдел:

```
SELECT dept.DNAME, Count(emp.EMPNO) AS CountOfEMPNO
FROM emp RIGHT JOIN dept ON emp.DEPTNO = dept.DEPTNO
GROUP BY dept.DNAME; /*Query34 */
```

Подреждане на кортежите

ORDER BY {ASC | DESC} позволява да се сортират кортежите по стойностите на атрибутите във възходящ или низходящ ред

```
SELECT ename, deptno, sal FROM emp ORDER BY deptno, sal DESC; /*
Query30 */
```

Обхождане на дървета

В случай на рефлексивно отношение (напр. Началник/подчинен или възел/детайл) се дефинира дървовидна структура, която е възможно да бъде обходена в ред – корен, поддървета, като се използва

SELECT [level] ... CONNECT BY PRIOR expr = expr START WITH expr

Тази конструкция не съществува в ACCESS и Interbase.

NUM	NOM	CHEF
1	A	NULL
2	D	NULL
3	H	2
4	B	1
5	E	2
6	F	5
7	C	1
8	G	5

SELECT Lpad(' **'**, 2*level)|| nom, nua FROM EMP CONNECT BY PRIOR cnef= num START WITH nom='A';

```
A 1
        B 4
        C 7
D 2
        E 5
                F 6
                G 8
        H 3
```

Пример от нашата БД:

```
SELECT LPAD(’ ’,2*(LEVEL-1)) || ename org_chart,empno, mgr, job
FROM emp
START WITH job = ’PRESIDENT’
CONNECT BY PRIOR empno = mgr;
```

Обновяване

Вмъкване

INSERT INTO table [(col1[,col2…])] VALUES(списък от стойности); или :

INSERT INTO table [(col1[,col2…])] VALUES SELECT ... ;

Пример :

INSERT INTO Emp (EMPNO, ENAME, JOB, HIREDATE, SAL, COMM, DEPTNO)

SELECT [EMPNO]+20 AS Expr5, ENAME, "CLERK" AS Expr2, #9/1/99# AS Expr4, 800 AS Expr3, COMM, 40 AS Expr1 FROM Emp WHERE DEPTNO=10;

Изменение

UPDATE table или view SET {column = expression | (list of columns)=(list of expressions)} [WHERE condition] ;

Изразът може да бъде SELECT оператор, който изработва необходимите стойности.

Пример:

UPDATE Emp SET Emp.SAL = [Sal]+100 WHERE DEPTNO=40;

Изтриване

DELETE FROM table [WHERE condition];

DELETE FROM emp WHERE DEPTNO=40**;**

Език за дефиниране на данните (DDL).

Създаване, премахване на таблици, изгледи, индекси

CREATE DATABASE за създаване БД (не и в ACCESS)

В Interbase съществува оператора :

CREATE DOMAIN за създаване на нови типове.

CREATE TABLE име (Attribut1 type1, Attribut2 type2, ...);

В ACCESS

CREATE TABLE table (field1 type [(size)] [NOT NULL] [index1] [, field2 type [(size)] [NOT NULL] [index2] [, ...]] [, CONSTRAINT multifieldindex [, ...]])

В Interbase

CREATE TABLE table [EXTERNAL [FILE] " <filespec>"]
(<col_def> [, <col_def> | <tconstraint> ...]);

<col_def> = col { datatype | COMPUTED [BY] (< expr>) | domain}
[DEFAULT { literal | NULL | USER}]
[NOT NULL] [<col_constraint>]
[COLLATE collation]

с типа може да се асоциира и клауза NOT NULL.

Друга форма на CREATE TABLE :

CREATE TABLE nom AS SELECT ....

DROP TABLE име; за премахване на таблицата

CREATE VIEW име [(имена на колони)] AS SELECT ....; Създаване на изглед ;

DROP VIEW име; премахване на изглед

Изгледът не съдържа физически данни. Съдържанието му се изчислява при пускане на заявките, които го използват. Иманта на атрибутите са нужни само в случай на промяна на имената на последнит по отношение на имената в участвуващите таблици или изгледи.

ALTER TABLE nom ADD(column_name type, ...); добавят се атрибути в таблица.

ALTER TABLE nom MODIFY(column type,. ...); модифицира типа на атрибут

ALTER TABLE nom DROP(column,. ...); изтрива атрибут

В Interbase не съществува клаузата MODIFY.

CREATE [UNIQUE] INDEX name_index ON name_table (attribut [ASCIDESC],...);

създава индекс за бърз достъп по един или повече атрибути в таблица

DROP INDEX nom_index; премахва индекса

Управление на правата на достъп

Общи права на достъп:

GRANT CONNECT | RESOURCE | ROLE | DBA(ADMIN) TO потребител IDENTIFIED BY парола;

CONNECT, RESOURCE са предефинирани роли в СУБД Oracle. Тяхното използване не се препоръчва в последната версия..

ROLE е именувано множество от привилегии които могат да се дават като едно цяло. То се обработва от командите:

CREATE ROLE

ALTER ROLE

SET ROLE

Достъп до обекти (с евентуално право за препредаване) :

GRANT SELECT | INSERT | DELETE | UPDATE | ALTER | INDEX | CLUSTER ON таблица или изглед TO потребител или роля [WITH GRANT OPTION] ;

GRANT OPTION дава на потребителя право да предава собствените си права на други потребители.

REVOKE отнема права на потребителите

REVOKE право ON таблица или изглед FROM потребител

Управление на транзакциите :

COMMIT записва резултата от транзакцията в БД.

ROLLBACK отменя промените направени от текущата транзакция и връща състоянието преди нейното начало.

Други елементи в СУБД

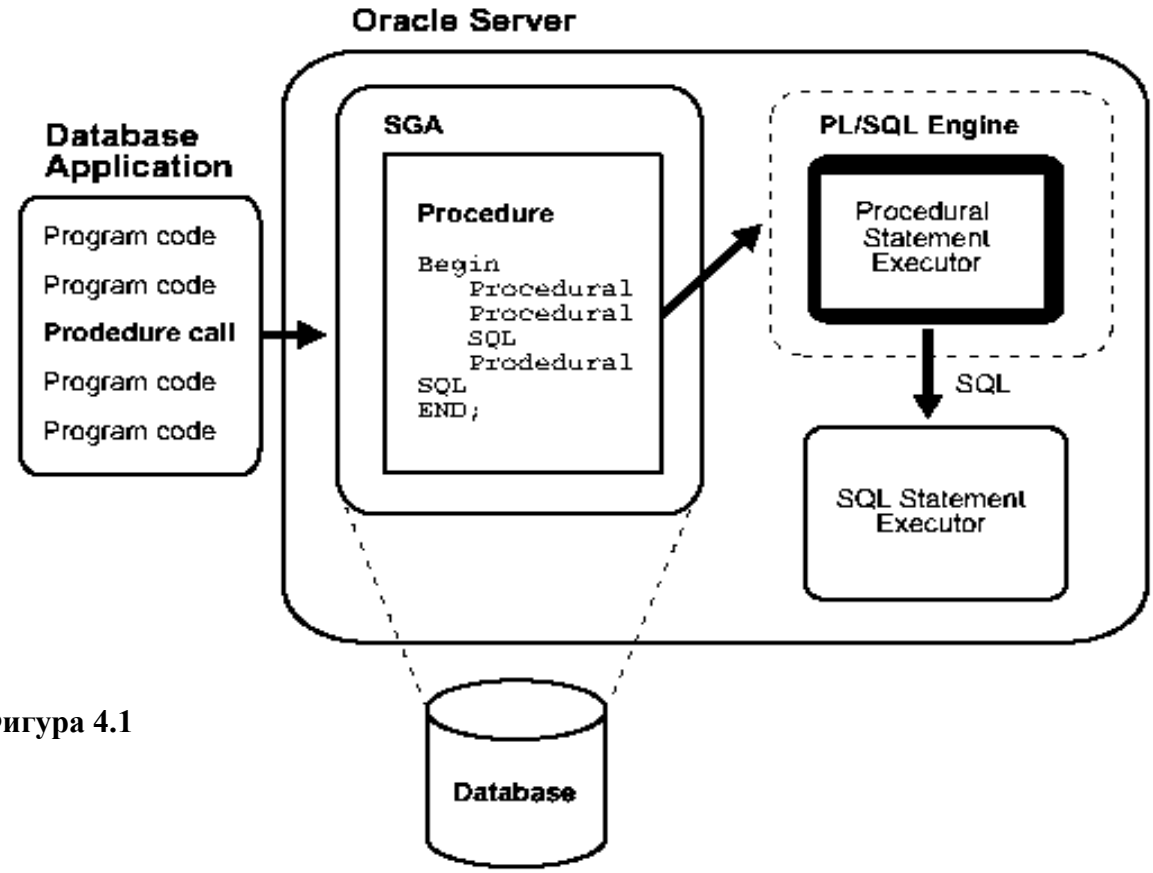
Следните елементи не съществуват в ACCESS

Курсор (Cursor)

Курсорите са указатели към съвкупност от редове разглеждани като записи, за да могат да бъдат обработвани един по един с помощта на алгоритмичен език. В ACCESS съществува обекта RECORDSET с тези свойства.

Съхранени процедури (Stored procedures)

Това са процедури и функции, написани на вграден алгоритмичен език в СУБД. Например такива са PL/SQL в ORACLE, Extended SQL в Interbase итн. Те се съхраняват заедно с БД и

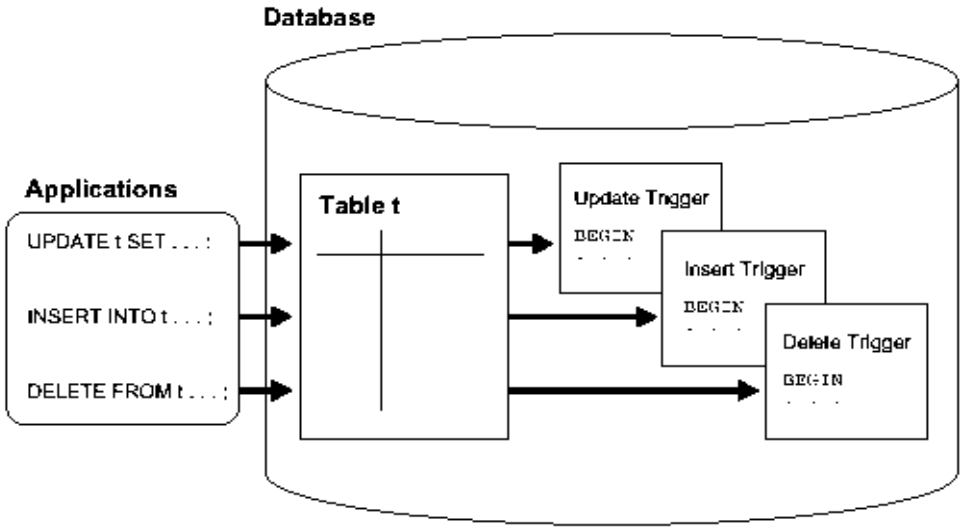


Фигура 4.1

могат да се използват от различни приложения. (Фиг. 4.1)

Тригери (Triggers)

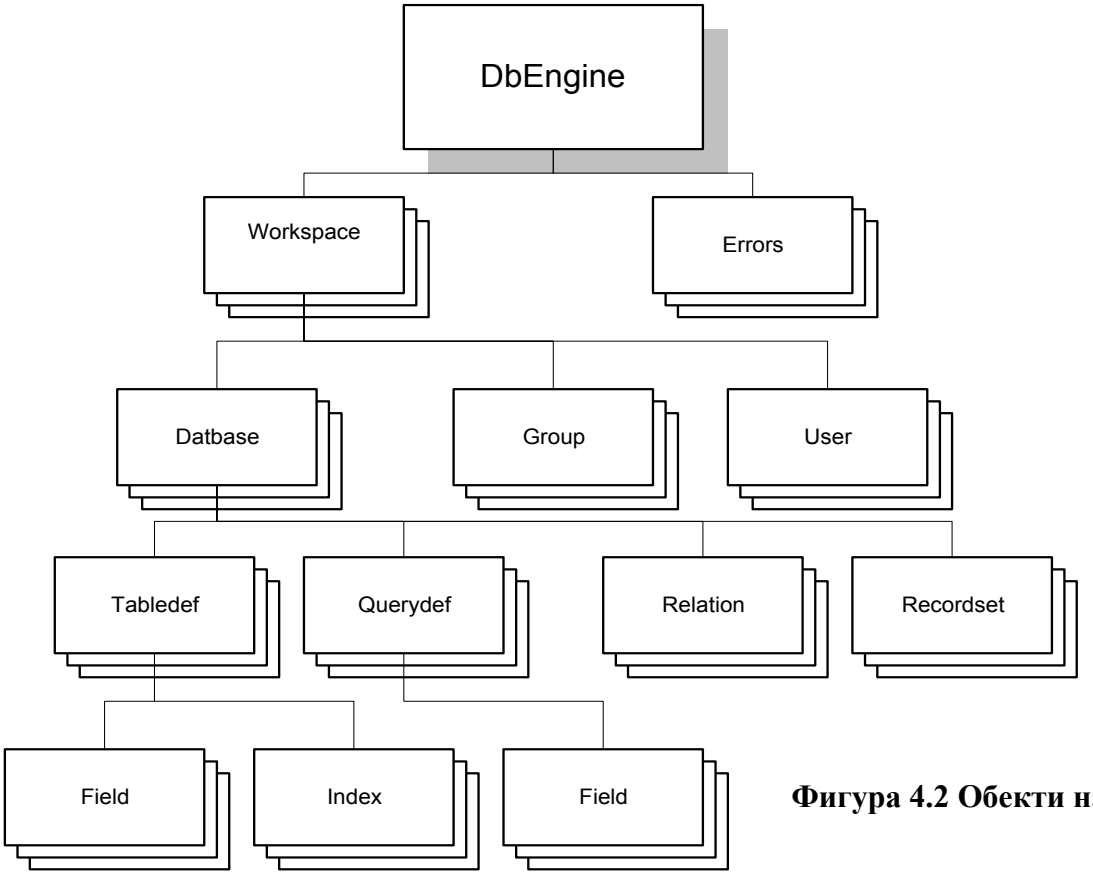
Това са процедури, които трябва да се изпълняват неявно, когато настъпи събитие от типа INSERT, UPDATE или DELETE. Те се използват, с цел да направят БД по-гъвкави. Могат да се изпълняват преди или след събитието и синтаксисът за виси от СУБД (Фиг. 4.2).



Фигура 4.2

Вградени езици

Освен SQL ние се нуждаем от процедурни езици, които описват алгоритми и процедури, които е невъзможно да се опишат чрез SQL. Всяка от СУБД има собствен език, който много прилича на някой от популярните езици (C, Pascal, PL, Basic). Например MS ACCESS използва VBA с обекти, които дават възможност за достъп до всички елементи на БД.



Фигура 4.2 Обекти на БД

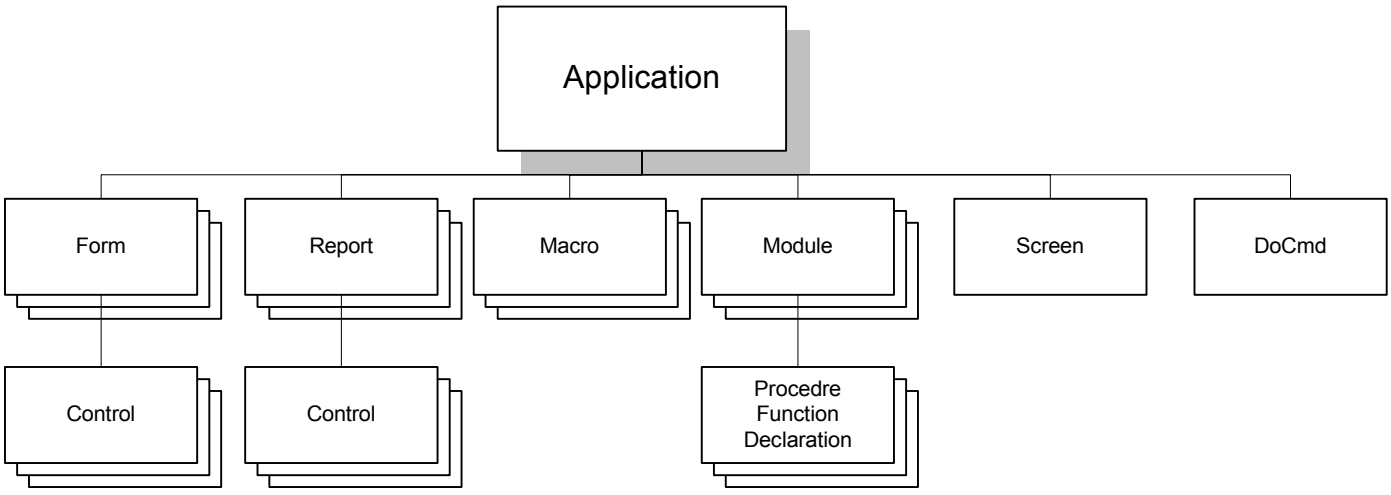


Figure 4.1 Приложни обекти

Йерархията на обектите е дадена на фигури 4.3 и 4.4. Всеки обект има свойства и методи, чрез които се осъществява достъпа и управлението.

Упражнения:

1. Да се намерят ФЗ и да се нормализира схемата, чиято реализация е:

A	*B*	*C*	*D*	*E*
a1	b1	c1	d1	e1
a2	b1	c1	d1	e4
a1	b1	c1	d2	e5
a1	b3	c3	d1	e2
a1	b1	c1	d1	e7
a1	b2	c1	d1	e3
a3	b1	c3	d1	e6
a4	b1	c1	d2	e7

Отговор : T1(A,D,E) , T2(A,B,C), T3(B,E)

2. Нека е дадена релационната схема:

Supplier(SupNo, Name, Country, Address, TelNo)

Article(ArtNo, Name, Unit)

Price(SupNo, ArtNo, PriceUn)

Delivery(LivNo, Date, SupNo)

Detail(LivNo, ArtNo, Quant)

Да се напишат операторите SQL Даващи отговор на следните въпроси:

1. Кои са доставчиците с имена съдържащи “ma ”?
2. Кои са стоките доставяни от американски доставчици?.
3. Какви са минималните, максималните и средните цени за всяка стока?
4. Какъв е броя на стоките доставян от всеки доставчик?
5. Кой доставчик не доставя никакъв вид сирене (всички имена на сирена съдържат името “сирене”)
6. Направете сравнение между средните цени на стоките доставяни от френски доставчици и средните цени на доставчиците от всички останали страни.
7. Направете месечен отчет по доставчици – всеки доставчик стоки на каква сума е доставил?
8. Направете месечен отчет по стоки (Стока, количество, сума)
9. Да се намери страната от която са внесени стоки на най-голяма сума
10. Намерете средната сума на дневните доставки за последния месец.