



KRÓTKI OPIS JĘZYKA SQL



Wrocław, styczeń 2003

W opracowaniu omówiono podstawowe informacje potrzebne do opanowania umiejętności zadawania pytań do baz danych SGM. Programy wchodzące w skład SGM wyposażone są w wiele standardowych raportów. Jednak żaden zestaw przewidzianych przez programistów pytań nie jest w stanie zaspokoić wszystkich potrzeb użytkowników systemu. W celu umożliwienia użytkownikom formułowania własnych pytań system SGM został wyposażony w możliwość obsługi pytań formułowanych w języku SQL.

1. Tabele SGM

Baza danych Systemu Gospodarki Mostowej została zrealizowana jako tak zwana „relacyjna baza danych”. Oznacza to (w dużym uproszczeniu), że informacje SGM zostały rozbite na pewną ilość grup a każda grupa informacji została zaś zrealizowana jako tabela (zwana również *relacją*). Wiersze tabel (zwane również *rekordami* lub *krotkami*) opisują pojedynczy zestaw danych (np. jeden obiekt mostowy). Kolumny tabel (zwane również *polami* lub *atrybutami*) opisują informacje związane z danym wierszem.

Przykład tabeli.

Główna tabela SGM nazwa się **JNIMain.db**. Oto jej fragment (ze sztucznymi danymi):

<i>JNI</i>	<i>LNI</i>	<i>Rodzaj</i>	<i>DrogaAdm</i>	<i>KmAdm</i>	<i>JAD</i>	<i>JAP</i>	<i>Miejscowo</i>	<i>...</i>
122212300	L231	JNI001a	123	23.341	WRWRWR	89887	Lubawka	...
122212301	L232	JNI001a	123	25.366	WRWRWR	89887	Mrągow	...
...

Cała tabela JNIMain.db zawiera 23 kolumny. Pełen ich wykaz z opisem znaczenia poszczególnych kolumn znajduje się w oddzielnym opracowaniu poświęconym strukturze tabel SGM.

Pierwsza kolumna tabeli JNIMain.db zawiera Jednolity Numer Inwentarzowy obiektu mostowego. Druga - Lokalny Numer Inwentarzowy. Kolejne dwie kolumny zawierają informacje o położeniu obiektu na drodze, do której obiekt jest przypisany administracyjnie. Następne kolumny zawierają kody Jednostki Administracji



Drogowej oraz Jednostki Administracji Państwowej na terenie, których leży obiekt, itd.

Jak widać na tym przykładzie pewne informacje zawarte w tej tabeli są zakodowane. Znaczenie pewnych kodów można wyczytać z dokumentacji SGM. Np. oto kody Rodzajów (kolumna nr 3):

- JN1001a most
- JN1001b tunel
- JN1001c przejście podziemne
- JN1001d przepust
- JN1001e prom

Jednostka Administracji Drogowej zapisana zaś jest w postaci sześciocyfrowego kodu. Znaczenie tego kodu zapisane jest zaś w tabeli o nazwie RdbJAD.db. W oddzielnej tabeli jest zapisane znaczenie kodu Jednostki Administracji Państwowej.

Tabele SGM zapisane są w formacie "Paradox 7". Jest jeden z powszechnie używanych standardowych formatów baz danych (innymi znanymi formatami są, np. dBase, InterBase, Sybase, Oracle,...). Tabele te mają rozszerzenie DB.

2. O języku SQL

Język **SQL** ("*Structure Query Language*" - "Strukturalny Język Zapytań"), rozwinięcie języka SEQUEL, jest językiem służącym do budowania systemów obsługi relacyjnych baz danych (RDBMS „*Relational DataBase Management System*”) na **dowolnej płaszczyźnie sprzętowej**. W chwili obecnej jest on standardowym językiem służącym do obsługi komercyjnych baz danych.

Programy napisane przez firmę ProMat służące do obsługi baz danych SGM wykorzystują jeden wspólny mechanizm dostępu do danych - "Borland Database Engine" (BDE). Z kolei BDE umożliwia dostęp do tabel baz danych za pomocą pewnej wersji języka SQL. Język ten jest podzbiorem standardu ANSI-92 SQL.

Język SQL umożliwia zadawanie pytań do baz danych oraz pozwala na zmienianie struktury baz danych. W opracowaniu tym omawiamy tylko ten fragment języka który służy do zadawania pytań.

Dokładny opis języka SQL można znaleźć w wielu książkach dostępnych na rynku.

UWAGA. Opracowanie to jest tylko wprowadzeniem do języka SQL. W żadnym wypadku nie zastępuje podręcznika. Wzorując się na podanych tutaj przykładach można nauczyć się tylko zadawania stosunkowo prostych pytań.

W przykładach będziemy posługiwali się tablicami:

1. **JN1Main.db** - tabela ta zawiera główne informacje o lokalizacji obiektów mostowych.



2. **MstMain.db** - tabela ta zawiera główne informacje o obiekcie, jeśli jest on mostem (a nie np. przepustem).
3. **MstPrzes.db** - tabela ta zawiera informacje o przęsłach mostów.
4. **JNIDrogi.DB** - tabela ta zawiera informacje o drogach przechodzących przez obiekt.

Pytania napisane w języku SQL nazywane są **zapytaniami**.

3. Elementy języka SQL

Oto najważniejsze zasady obowiązujące przy formułowaniu zapytań w języku SQL:

1. W każdym zapytaniu występują tzw. klauzule rozpoczynające się od słów kluczowych (np. SELECT, FROM, WHERE...).
2. Zapytania SQL mogą być zapisane w jednym lub kilku wierszach.
3. Małe i wielkie litery nie są rozróżniane. Nie dotyczy to jednak wartości pisanych w cudzysłowach.

3.1. Proste zapytania

Najprostsze zapytanie ma postać:

```
SELECT *  
FROM tabela
```

Przykład 1.

```
SELECT *  
FROM "JNIMain.DB"
```

Zostaną wybrane wszystkie obiekty mostowe (i ich wszystkie dane) zapisane w tabeli JNIMain.DB.

Aby wyświetlić jedynie część informacji z tabeli w zapytaniu należy wypisać nazwy kolumn, które mają zostać wyświetlone:

```
SELECT wyrażenie_1 [, wyrażenie_2, ..., wyrażenie_n]  
FROM tabela [tabela_2, ..., tabela_n]
```

W wyrażeniach możemy stosować nazwy kolumn oraz operatory arytmetyczne: +, -, *, /, operator konkatencji || (łączenie napisów).

Przykład 2.

```
SELECT
```



```
JNI,  
Rodzaj,  
DrogaAdm,  
KmAdm  
FROM "JNIMain.DB"
```

Zostaną wyświetlone numery JNI, rodzaje obiektów, numery dróg i kilometrów wszystkich obiektów mostowych zapisanych w tabeli JNIMain.DB.

Jedną z zalet zapytań SQL jest możliwość nakładania warunków na dane zwracane w wyniku zapytania. Podstawowym mechanizmem stosowanym dla narzucania ograniczeń na rekordy zwracane w wyniku zapytań jest klauzula WHERE.

```
SELECT wyrażenie_1 [, wyrażenie_2, ..., wyrażenie_n]  
FROM relacja  
WHERE warunek_1 [AND/OR warunek_2 ... AND/OR warunek_n]
```

W warunkach występujących w klauzuli WHERE stosuje się zwykle operatory arytmetyczne:

Symbol	Znaczenia
=	równość
<>	nierówność
>	większość
>=	większość lub równość
<	mniejszość
<=	mniejszość lub równość

oraz operatory języka SQL

Operator	Znaczenie
between ... and ...	wartość z przedziału
like	wartość zgodna z wzorcem
is null	wartość pusta - null
in	wartość w zbiorze danych

Każdy z operatorów SQL może być poprzedzony negacją **not**, która zmienia jego logiczne znaczenie. Warunki złożone zbudowane są z warunków prostych powiązanych spójnikami logicznymi AND i OR.

Przykład 3.

```
SELECT *  
FROM "JNIMain.DB"
```



```
WHERE Rodzaj = 'JNI001a'
```

Zostaną wyświetlone wszystkie informacje o obiektach, które są mostami. Przypomnijmy, że w tabeli „JNIMain.db” kod JNI001a oznacza most, JNI001b -tunel, JNI001c- przejście podziemne, JNI001d– przepust oraz JNI001e-prom.

Wyniki zapytań generowane są w postaci tabel, gdzie poszczególne wiersze reprezentują kolejne obiekty uzyskiwane w wyniku zapytania. Kolumny w tabeli stanowiącej wynik zapytania opatrzone są nagłówkiem, który jest opisem wyrażenia, którego wynik pojawia się w kolejnych wierszach w tej kolumnie. Niekiedy taka postać nie jest najbardziej eleganckim i czytelnym sposobem na nazwanie kolumny pojawiającego się wydruku i często dla kolumny w tabeli nadajemy osobny alias (nazwę zastępczą), który występuje jako nagłówek kolumny wyników zapytania.

```
SELECT atrybut_1 AS Atrybut_Jeden, atrybut_2 AS Atrybut_Dwa ...  
FROM relacja  
[WHERE warunki]
```

Przykład 4.

```
SELECT  
JNI,  
Rodzaj AS Rodzaj_Obiektu,  
DrogaAdm AS DROGA_NR,  
KmAdm AS Km  
FROM "JNIMain.DB"
```

Zostaną wyświetlone numery JNI, numery dróg i kilometrów wszystkich obiektów mostowych zapisanych w tabeli JNIMain.DB. Kolumny zostaną nazwane (w kolejności) JNI, Rodzaj_Obiektu, DROGA_NR, Km.

Celem unikania wielokrotnych wystąpień identycznych wierszy w tabeli stanowiącej wynik zapytania można zastosować klauzule DISTINCT.

```
SELECT DISTINCT atrybut  
FROM tabela  
[WHERE warunek]
```

Przykład 5.

```
SELECT  
DISTINCT(JAD)  
FROM "JNIMAIN.DB"  
WHERE Rodzaj = JNI001e;
```



Zostaną wyświetlone kody JAD jednostek drogowych, w których występują promy (promy mają kod JN1001e). Jeżeli w danej jednostce występuje wiele obiektów to i tak dany kod zostanie wypisany tylko raz.

Wyniki zapytań zwracane są w kolejności ich występowania w tabelach lub w kolejności określonej indeksami. Jest jednak możliwość sortowania generowanych wyników. Sortowanie odbywa się rosnąco (ustawienie domyślnie) lub malejąco. Sposób sortowania określa słowo **ASC** (ascending - rosnąco) lub **DESC** (descending - malejąco) W przypadku sortowania wyników zapytania mamy także możliwość podania atrybutu (lub grupy atrybutów) wg których proces ten będzie realizowany. W przypadku określenia kilku kryteriów sortowania czyli np. kilku nazw kolumn sortowanie odbywa się najpierw wg pierwszego wymienionego kryterium, a w sytuacji wystąpienia takich samych wartości o kolejności wystąpienia obiektów w wyniku decydują wartości kolejnych kluczy sortujących. Oto format zapytania z określonym porządkiem:

```
SELECT atrybut_1, atrybut_2, atrybut_3
FROM tabela
[WHERE warunki]
ORDER BY "Atrybut jeden", „atrybut_dwa”, „atrybut_3” [ASC|DESC]
```

Przykład 6.

```
a) SELECT
    JN1,
    Rodzaj,
    DrogaAdm,
    KmAdm
FROM "JN1MAIN.DB"
WHERE Rodzaj = 'JN1001d'
ORDER BY DrogaAdm, KmAdm
```

```
b) SELECT
    JN1,
    Rodzaj,
    DrogaAdm,
    KmAdm
FROM "JN1Main.DB"
WHERE Rodzaj = 'JN1001d'
ORDER BY DrogaAdm, KmAdm DESC
```

Zostaną wyświetlone wszystkie przepusty (kod=JN1001e), będą one uszeregowane według

- a) danej drogi i rosnącego pikietarzu,
- b) danej drogi i malejącego pikietarzu.



Ogólna postać zlecenia SELECT ze wszystkimi poznanymi dotąd klauzulami wygląda następująco:

```
SELECT [DISTINCT] { *, wyrażenie [alias], ... }  
FROM tabela  
WHERE warunki  
ORDER BY { atrybut, wyrażenie } [ASC|DESC]
```

Wyniki zapytania pochodzić mogą z wielu tabel. Jeżeli w zapytaniu następuje odwołanie do atrybutów pochodzących z wielu tabel, przy czym nazwy atrybutów w tych relacjach mogą się powtarzać to dla jednoznaczności zapytania należy zastosować aliasy (nazwy zastępcze) dla relacji. Aliasy specyfikujemy za nazwą tabeli (relacji), zaś odwołania do atrybutów z poszczególnych relacji poprzedzamy nazwą aliasu oddzielonego kropką od nazwy atrybutów. Oto format zapytania z aliasami:

```
SELECT r1.atribut_1, r1.atribut_2, r2.atribut_1  
FROM tabela_1 r1, tabela_2 r2  
[WHERE warunek]
```

Przykład 7.

```
SELECT *  
FROM "JniMain.db" J, "MstMain.db" M, "MstPrzes.db" P  
WHERE (J.Jni = M.Jni) AND (J.JNI=P.JNI) AND (P.PoMaKon='MST079i');
```

Zostaną wyświetlone obiekty, w których przynajmniej jeden materiał konstrukcji pomostu jest wykonany z drewna.

3.2. Podzapytania

Podzapytanie jest pytaniem zbudowanym z klauzuli SELECT, które pojawia się w ramach innego zapytania. Niekiedy nazywane jest zapytaniem zagnieżdżonym. Wynik podzapytania staje się częścią zapytania nadrzędnego. Podzapytanie, jak każde zapytanie, może zawierać inne podzapytania.

```
SELECT lista_select  
FROM relacja_1 alias_1  
WHERE wyr_1 operator (SELECT wyr_2  
FROM relacja_2 alias_2  
WHERE warunek)
```

Przykład 8.

```
SELECT  
J.JNI  
FROM "JniMain.db" J
```



```
WHERE J.Jni IN ( SELECT
                D.JNI
                FROM "JNIDrogi.DB" D
                WHERE
                D.Droga = 'A4')
```

Zostaną wyświetlone wszystkie obiekty w ciągu drogi A4.

3.3. Funkcje agregujące

Funkcje agregujące służą do uzyskiwania informacji statystycznych np. liczba mostów na danej drodze. Dziedziną tych funkcji są grupy. Na grupy składają się krotki (rekordy, obiekty), które spełniają zadany warunek. Do grupowania rekordów służy klauzula GROUP BY. Funkcje agregujące występujące w języku SQL to:

1. MIN – do wyznaczania minimum,
2. MAX – do wybrania maksymalnego elementu,
3. SUM – do sumowania wartości danej kolumny,
4. COUNT – do zliczania liczby obiektów spełniających warunek podany w klauzuli WHERE,
5. AVG – do wyznaczania średniej wartości kolumny.

Przykład 9.

```
SELECT
    DrogaAdm,
    COUNT(JNI) AS "Liczba_obiektów"
FROM "JniMain.db"
GROUP BY DrogaAdm
```

Zostaną wyświetlone liczby obiektów na danych drogach.

4. SŁOWA KLUCZOWE JĘZYKA SQL

1. ACTIVE, ADD, ALL, AFTER, ALTER, AND, ANY, AS, ASC, ASCENDING, AT, AUTO, AUTOINC, AVG
2. BASE_NAME, BEFORE, BEGIN, BETWEEN, BLOB, BOOLEAN, BOTH, BY, BYTES
3. CACHE, CAST, CHAR, CHARACTER, CHECK, CHECK_POINT_LENGTH, COLLATE, COLUMN, COMMIT, COMMITTED, COMPUTED, CONDITIONAL, CONSTRAINT, CONTAINING, COUNT, CREATE, CSTRING, CURRENT, CURSOR
4. DATABASE, DATE, DAY, DEBUG, DEC, DECIMAL, DECLARE, DEFAULT, DELETE, DESC, DESCENDING, DISTINCT, DO, DOMAIN, DOUBLE, DROP
5. ELSE, END, ENTRY_POINT, ESCAPE, EXCEPTION, EXECUTE, EXISTS, EXIT, EXTERNAL, EXTRACT
6. FILE, FILTER, FLOAT, FOR, FOREIGN, FROM, FULL, FUNCTION
7. GDSCODE, GENERATOR, GEN_ID, GRANT, GROUP, GROUP_COMMIT_WAIT_TIME
8. HAVING, HOUR



9. IF, IN, INT, INACTIVE, INDEX, INNER, INPUT_TYPE, INSERT, INTEGER, INTO, IS, ISOLATION
10. JOIN
11. KEY
12. LONG, LENGTH, LOGFILE, LOWER, LEADING, LEFT, LEVEL, LIKE, LOG_BUFFER_SIZE
13. MANUAL, MAX, MAXIMUM_SEGMENT, MERGE, MESSAGE, MIN, MINUTE, MODULE_NAME, MONEY, MONTH
14. NAMES, NATIONAL, NATURAL, NCHAR, NO, NOT, NULL, NUM_LOG_BUFFERS, NUMERIC
15. OF, ON, ONLY, OPTION, OR, ORDER, OUTER, OUTPUT_TYPE, OVERFLOW
16. PAGE_SIZE, PAGE, PAGES, PARAMETER, PASSWORD, PLAN, POSITION, POST_EVENT, PRECISION, PROCEDURE, PROTECTED, PRIMARY,
17. PRIVILEGES
18. RAW_PARTITIONS, RDB\$DB_KEY, READ, REAL, RECORD_VERSION, REFERENCES, RESERV, RESERVING, RETAIN, RETURNING_VALUES, RETURNS, REVOKE, RIGHT, ROLLBACK
19. SECOND, SEGMENT, SELECT, SET, SHARED, SHADOW, SCHEMA, SINGULAR, SIZE, SMALLINT, SNAPSHOT, SOME, SORT, SQLCODE, STABILITY, STARTING, STARTS, STATISTICS, SUB_TYPE, SUBSTRING, SUM, SUSPEND
20. TABLE, THEN, TIME, TIMESTAMP, TIMEZONE_HOUR, TIMEZONE_MINUTE, TO, TRAILING, TRANSACTION, TRIGGER, TRIM
21. UNCOMMITTED, UNION, UNIQUE, UPDATE, UPPER, USER
22. VALUE, VALUES, VARCHAR, VARIABLE, VARYING, VIEW
23. WAIT, WHEN, WHERE, WHILE, WITH, WORK, WRITE
24. YEAR

OPERATORY

||, -, *, /, <>, <, >, =, <=, >=, ~=, !=, ^=, (,)

Operatory ~=, !=, ^= oznaczają „różne”. Operator || służy do złączenia dwóch napisów.

5. Przykłady firmy Borland

W rozdziale tym przytoczymy przykłady znajdujące się w pomocniku „Database DeskTop User’s Guide” do programu „DataBase Desktop 7” firmy Borland. Pierwsze trzy przykłady dotyczą metod zmiany danych w tabelach za pomocą języka SQL. Narzędzia systemu SGM nie pozwalają użytkownikowi na korzystanie z tej grupy poleceń. Przykłady te zamieszczamy tylko ze względu na zupełność przeglądu poleceń SQL.

5.1. Przykłady zapytań

Język zadawania pytań zawiera następujące wyrażenia:

SELECT FROM, WHERE, ORDER BY, GROUP BY, and HAVING

Do agregowania danych możesz korzystać z następujących funkcji:

SUM, AVG, MIN, MAX, COUNT

Posługiwać się możesz następującymi operatorami:

+, -, *, /, =, <>, IS NULL, IS NOTNULL, >=, =<, AND, OR, NOT, ||, LIKE



Następujące przykłady ilustrują typowe zapytania do niezbyt złożonych baz danych:

Przykład 1: UPDATE

```
update produkty
set miasto = 'Wrocław'
where produkty.miasto = 'Warszawa'
```

Przykład 2: INSERT

```
insert into produkty (IdP, miasto)
values ( 'aa0094', 'Wrocław' )
```

Przykład 3: DELETE

```
delete from produkty
where IdP = 'aa0093'
```

Przykład 4: SELECT (użyte do złączenia table)

W przykładzie tym wykorzystane jest polecenie SELECT do złączenia danych z dwóch różnych tabel:

```
select distinct
  A.IdP, A.ilosc, B.miasto
from magazyn A, produkty B
where
  A.IdP = B.IdP and B.ilosc > 20
order by A.ilosc, B.miasto, A.IdP
```

Formuła $A.IdP = B.IdP$ w powyższym pytaniu służy do złączenia table magazyn oraz produkty.

Przykład 5: PODZAPYTANIA

W pytaniach mogą być zawarte podzapytania:

```
select p.part_no
from parts p
where
  p.quantity in (select i.quantity
                 from inventory i
                 where i.part_no = 'aa9393')
```

Przykład 6: GROUP BY

Oto prosty przykład zapytania grupującego:

```
select
  part_no, sum(quantity) as PQTY
from parts
group by part_no
```

Uwaga: Wszystkie pola niegrupowane w części SELECT (w naszym przypadku pole part_no) muszą występować po klauzuli GROUP BY.

Przykład 7: ORDER BY



Następujący przykład pokazuje wykorzystanie DESCENDING w poleceniu ORDER BY:

```
select distinct
  customer_no
from c:\data\customer
order by customer_no descending
```

5.2. Grupowanie

Oto lista wszystkich funkcji, które mogą być użyte w pytaniach grupujących:

- SUM(), suma wszystkich wartości numerycznych w danym polu
- AVG(), wartość średnia wszystkich niepustych wartości
- MIN(), minimalna wartość pól w danej kolumnie
- MAX(), maksymalna wartość pól w danej kolumnie
- COUNT(*), for counting non-NULL numeric values in a column

W wyrażeniach grupujących mogą występować formuły:

- SUM(Field * 10)
- SUM(Field) * 10
- SUM(Field1 + Field2)

5.3. Funkcje łańcuchowe

Oto lista funkcji operujących na napisach, które mogą być używane do wyszukiwania, wstawiania i modyfikacji:

- UPPER(), podniesienie napisu do dużych liter,
- LOWER(), zamiana wszystkich liter w napisie na małe litery,
- TRIM(), usunięcie wskazanego znaku z przodu, z tyłu lub z obu stron napisu,
- SUBSTRING() wycinanie podnapisu z danego napisu.

5.4. Funkcje operujące na datach

Do wydobywania informacji z pola typu "DATE" można posłużyć się następującą funkcją:

```
EXTRACT (typ_informacji FROM nazwa_kolumny)
```

Typem informacji może być: YEAR (rok), MONTH (miesiąc), DAY (dzień), HOUR (godzina), MINUTE (minuta) i SECOND (sekunda). Następujący przykład pokazuje jak można wydobyć z daty rok:

```
SELECT
  EXTRACT(YEAR FROM HIRE_DATE)
FROM EMPLOYEE;
```