1 SQL – uzupełnienie

W przykładach do tej części będziemy rozważać następującą tabelę o nazwie `emp`:

id	name	position	sal	manager	dept	login
1367	"Marek"	"Operator serwera"	1200	1369	1	"pracownik1"
1368	"Agata"	"Programista"	1400	1369	1	"pracownik10"
1369	"Aneta"	"Manager zespołu"	2500	NULL	1	"manager2"
1370	"Zbigniew"	"Dział Kadr"	1500	NULL	2	"pracownik3"
1371	"Weronika"	"CEO"	4000	NULL	2	"pracownik"
1372	"Arkadiusz"	"Programista"	1300	1369	1	"pracownik123"

1.1 Operator AS w nazwach kolumn

Na poprzednich zajęciach używaliśmy operatora `AS`, żeby nadać alias dla tabeli, ale możemy to także zrobić dla kolumn. Pomaga to nam na przykład, kiedy używamy funkcji agregujących:

- baza danych automatycznie nadaje nazwy kolumnom wynikowym – jeśli używamy funkcji agregujących, np. `COUNT`, to zostanie ona nazwana podobnie do `COUNT_OF_DEPT`, co nie wygląda zbyt ładnie, jeśli ma to być jakieś oficjalne zestawienie (może też być niewygodne, jeśli chcemy napisać sobie jakiś skrypt na bazie danych i potem np. zmieni się nazwa kolumny – będziemy musieli ją zmieniać w wielu miejscach)
- jeśli chcemy użyć agregacji w warunku `WHERE` i jednocześnie wypisać wartość tej agregacji w wynikowej tabeli i nie chcemy tego pisać dwa razy

Przykład:

```
SELECT a.dept, COUNT(a.dept) AS liczba_prac
FROM emp AS a
WHERE liczba_prac > 2
GROUP BY a.dept
```

1.2 Operator LIKE

Często w SQLu będziemy chcieli wyszukiwać stringi, które pasują do jakiegoś wzorca, dlatego w standardzie zawarty jest operator `LIKE`, który wspiera bardzo podstawowe funkcje regexa. Dwa wildcardy, z których będziemy korzystać, to `%` i _.

`%` – reprezentuje 0 lub więcej znaków; przykład zastosowania:

```
SELECT login
FROM emp
WHERE login LIKE "pracownik%"
```

Wówczas w odpowiedzi dostaniemy tabelę zawierającą loginy: `pracownik1, pracownik10, pracownik3, pracownik, pracownik123`.

_ – reprezentuje dokładnie jeden znak; przykład zastosowania:

```
SELECT login
FROM emp
WHERE login LIKE "pracownik_"
```

Wtedy odpowiedź będzie wyglądała następująco: `pracownik1, pracownik3`.

Możemy też połączyć oba te operatory, żeby np. wymusić przynajmniej `n` znaków, ale dopuścić więcej. W poniższym przykładzie szukamy pracowników, którzy w loginie mają co najmniej dwie cyfry:

```
SELECT login
FROM emp
WHERE login LIKE "pracownik__%"
```

W odpowiedzi dostajemy: `pracownik10, pracownik123`.

1.3 Operacje na czasie

Składnia dotycząca czasu, dostępne typy i funkcje różnią się (często znacząco) między różnymi implementacjami baz danych, więc dzisiaj skupimy się na funkcjonalności dostępnej w Microsoft Access (program, z którym będziemy pracowali).

- `Date()` – zwraca aktualną datę
- `Now()` – zwraca aktualny czas (czyli datę + godzinę)
- `Time()` – zwraca aktualny czas sformatowany jako string
- `DateAdd(interval, number, date)` – dodaje do daty odpowiedni czas zależny od `interval` (np. `s` – sekundy, `n` – minuty, `m` – miesiące, pełną listę możliwych jednostek czasu można znaleźć w internecie)
- `DatePart(interval, date, [firstdayofweek], [firstweekofyear])` – zwraca wartość podanej jednostki czasu; UWAGA! niedziela jest domyślnie pierwszym dniem tygodnia
- `Minute(date), Day(date), Month(date), Year(date), Weekday(date)` – podobnie jak `DatePart` i ta sama uwaga!
- `DateDiff(interval, date1, date2, [firstdayofweek], [firstweekofyear])` – zwraca różnicę między datami w podanej jednostce czasu

Na czasie można też wykonywać normalne operacje arytmetyczne +/-, ale nie zawsze będą się one dobrze zachowywały – jeśli to możliwe, lepiej użyć funkcji.

Pełna lista funkcji dostępna w dokumentacji.

1.4 Zwracanie wyrażenia arytmetycznego

W wyrażeniu `SELECT` można oczywiście wykonywać standardowe operacje arytmetyczne, w których argumentami mogą być liczby lub kolumny. Kilka przykładów:

- wypisujemy dodatkową kolumnę, która jest liczbą:
  `SELECT name, sal, 100 AS nazwa FROM emp`
- sprawdzamy, co by się stało, gdyby wszyscy dostali podwyżkę:
  `SELECT name, sal, sal + 100 AS nowa_pensja FROM emp`
- podwyżka zależna od departamentu, w którym ktoś pracuje:
  `SELECT name, sal, sal + dept * 100 AS nowa_pensja FROM emp`

1.5 Instrukcje warunkowe

Można w wyrażeniach `SELECT` używać instrukcji warunkowych `IF` i `CASE` – ich składnia znowu będzie się różniła między wariantami SQLa. Przykłady w miarę standardowe:

```
SELECT name, sal, IF (dept == 1) THEN sal + 100 ELSE sal END FROM emp
```

```
SELECT name, sal, CASE dept WHEN 1 THEN sal + 100 WHEN 2 THEN sal END FROM emp
```

2 Praca z bazą danych

Na zajęciach będziemy korzystać z programu Microsoft Access, który jest dostępny w ramach pakietu Office Professional.

2.1 Zadanie maturalne – uczniowie

Dane do zadania i treść można pobrać na stronie internetowej przy dacie dzisiejszych zajęć.