Projekt 1. Baza danych

Praktyka Programowania 2013/2014; Wydział ETI Politechniki Gdańskiej

Spis treści

1. Wprowadzenie

Projekt polega na stworzeniu prostej bazy danych, z którą komunikacja odbywa się tekstowo. Funkcje, które w jakimś zakresie Państwa program będzie musiał implementować to dodawanie danych, ich wyświetlanie, usuwanie oraz odczyt/zapis do plików. Poniżej znajduje się skrócona lista funkcjonalności do implementacji – te wypisane pogrubioną czcionką są wymagane, natomiast implementacja pozostałych zależy od Państwa umiejętności oraz włożonej pracy.

1. Obsługa typów danych (i odpowiadających im predefiniowanych tabel) [3 pkt.]
    a. **obsługa int i string, trzy tabele [1]**
    b. obsługa int, string, double, bool, pięć tabel [1]
    c. wybrany[1] z atrybutów kolumny
        i. autoincrement [1], albo
        ii. ograniczenie wstawianych wartości int [0.5]
2. Dodaj [2 pkt.]
    a. **dodawanie całych wierszy [1]**
    b. wybrany[2] z dwóch:
        i. dodawanie wybranych komórek w dowolnej kolejności [1], albo
        ii. dodawanie wielu rekordów równocześnie [0.5]
3. Wypisz [5 pkt.]
    a. **wyświetlanie całej zawartości posortowanej po kolejnych polach wszystkich tablic (domyślny tryb) [1]**
    b. rozszerzenia do realizacji na dwóch tabelach
        i. wybór określonych kolumn do wyświetlenia [1]
        ii. określenie zakresu wyświetlania [0.5]
        iii. sortowanie [1]
        iv. kryterium wyboru: ograniczony warunek [0.5]
        v. kryterium wyboru: dowolny warunek [1]
4. Skasuj [3 pkt.]
    a. **pojedynczy warunek we wszystkich tabelach [1]**
    b. obsługa wszystkich atrybutów [2]
5. Zapis i odczyt [2 pkt.]
    a. **zapis i odczyt wszystkiego co jest we wszystkich tabelach [1]**
    b. zapis wybranej tabeli do pliku określonego w poleceniu, odczyt ma sprawdzić czy to odpowiednia tabela [1]

Wszystkie te funkcjonalności są szczegółowo opisane w dalszej części dokumentu.

1.1. Środowisko

Program powinien zostać zrealizowany w środowisku MS Visual C++ w wersji zgodnej z tą dostępną w laboratorium. Zadanie jest zdefiniowane w taki sposób, że można je zrealizować w samym języku C, jeśli jednak ktoś już programuje obiektowo – nic nie stoi na przeszkodzie, żeby podejść do projektu w ten sposób. Jednak warto tu podkreślić, że o ile uporządkowanie swojego kodu w formie klas (jeśli jest zrobione poprawnie) jest lepszym rozwiązaniem niż prosty zestaw funkcji realizujący podobną funkcjonalność to w ramach tego projektu ten aspekt nie będzie uwzględniany w ocenie (na to przyjdzie czas w drugim semestrze).

[1] to znaczy jeden z dwóch do wyboru, implementacja obu nie da dodatkowych punktów
[2] jak wyżej, maksymalnie 1 punkt

Program ma działać w trybie tekstowym – przyjmować polecenie ze standardowego wejścia, i wypisywać odpowiedzi na standardowe wyjście. Tutaj od razu wskazówka – do testów programu warto zapoznać się z przekierowywaniem wejścia oraz wyjścia programu przy uruchamianiu w wierszu poleceń.

Można przyjąć, że polecenia będą wprowadzane bezbłędnie – więc program nie musi sprawdzać ich poprawności. Każde polecenie będzie wprowadzane w osobnej linii, żadna linia we wprowadzanym zestawie poleceń nie będzie dłuższa niż 4096 znaków. Środowiskiem uruchomieniowym będzie MS Windows, więc kolejne linie wejścia rozdzielają dwa znaki.

Funkcjonowanie programu będzie częściowo sprawdzane automatycznie, więc Państwa programy muszą ściśle przestrzegać opisanego w tym dokumencie formatu wyjścia. Informacje na temat sposobu testowania oraz zgłaszania programów do oceny będą opisane w osobnym dokumencie.

1.2. Krótki przykład

Poniższa tabela zawiera przykładową komunikację z programem – po lewej mamy wprowadzane instrukcje – po prawej odpowiedzi.

Wejście programu	Wyjście programu
`dzien dobry`	`?`
`dodaj liczby (1)`	`Liczba dodanych rekordow: 1`
`dodaj liczby (5)`	`Liczba dodanych rekordow: 1`
`dodaj liczby (3)`	`Liczba dodanych rekordow: 1`

`wypisz liczby`	`Liczba rekordow: 3`
	`wartosc`
	`1`
	`3`
	`5`

`wypisz liczby gdzie wartosc>2`	`Liczba rekordow: 2`
	`wartosc`
	`3`
	`5`

`usun liczby gdzie wartosc<4`	`Liczba usuniętych rekordow: 2`

`wypisz liczby`	`Liczba rekordow: 1`
	`wartosc`
	`5`

`dodaj studenci (1234,”Jan”,”Now”)`	`Liczba dodanych rekordow: 1`
`dodaj studenci (1235,”Ada”,”Now”)`	`Liczba dodanych rekordow: 1`
`dodaj studenci (1236,”Jan”,”Kan”)`	`Liczba dodanych rekordow: 1`

`wypisz studenci wedlug nazwisko`	`Liczba rekordow: 3`
	`indeks imie nazwisko`
	`1236 "Jan" "Kan"`
	`1234 "Jan" "Now"`
	`1235 "Ada" "Now"`[3]

`koniec`	

Uwaga: wejście i wyjście programu zostało podzielone na dwie kolumny dla czytelności. Program na w wejściu, jak i wyjście nie ma żadnych

[3] Dlaczego taka kolejność? Bo podanie samego nazwiska oznacza sortowanie kolejno po nazwisku, numerze indeksu i imieniu. O tym dalej…

2. Szczegółowy opis zadania

W tej części znajdą Państwo dokładny opis wszystkich funkcjonalności programu. W przypadku wątpliwości należy skierować się do osób opiekujących się laboratoriami otwartymi, jeśli okaże się, że treść zadania wymaga doprecyzowania – to te osoby przekażą informację autorowi zadania.

2.1. Definicja tabel

Prawdziwe DBMSy mają zestaw poleceń pozwalających na definiowanie własnych tabel – my przyjęliśmy pewne uproszczenie polegające na dokładnym określeniu jakie tabele mają być obsługiwane. Co więcej, określamy maksymalny rozmiar tabeli na 1000 – przy próbie dodania kolejnego rekordu – należy poinformować użytkownika o niepowodzeniu operacji w odpowiednim komunikacie.

Program musi zapewniać obsługę trzech tabel:

nazwa	*atrybuty*
liczby	(int wartosc)
studenci	(int indeks; string imie; string nazwisko)
przedmioty	(*auto* int id; string nazwa; *clamp (1,10)* int semestr)

Atrybuty (albo kolumny) tabeli określone są przez typ (o nich więcej poniżej) oraz nazwę atrybutu. Dodatkowo w definicji tabeli `przedmioty` przy atrybutach `id` oraz `semestr` znajdują się dodatkowe modyfikatory. Oznaczają one odpowiednio atrybut, na którym należy zaimplementować autoinkrementację (`auto`) oraz atrybut z ograniczonym zakresem wartości (`clamp`). To, w jaki sposób mają one funkcjonować opisane jest w części poświęconej dodawaniu rekordów.

Aby uzyskać wyższą punktację należy rozszerzyć program o obsługę jeszcze jednej tabeli, w której wykorzystane są dwa dodatkowe typy danych:

sale	(string nazwa, *clamp(10, 600)* int rozmiar, bool projektor, double powierzchnia)

2.1.1. Typy danych

Stworzone przez Państwa programy powinny obsługiwać następujące typy danych:

typ	domyślnie	uwagi
string	""	do 255 znaków (z zakresu ASCII 33..127, poza 34, 40, 41(cudzysłów i zwykłe nawiasy)); dla stringów muszą działać wszystkie operatory (>, < porównują alfabetycznie z uwzględnieniem wielkości liter); zawsze w cudzysłowach
int	0	standardowy zakres int32
double	0.000	format wejścia: separator dziesiętny – kropka, format wyjścia: zawsze dokładnie trzy cyfry po kropce,
bool	falsz	prawda/falsz, (fałsz); bez cudzysłowów; nie ma potrzeby obsługi operatorów >,<

2.2. Przetwarzanie poleceń i zakończenie pracy programu

Pierwszy etap jest najłatwiejszy – należy napisać program, który będzie oczekiwał na wprowadzanie kolejnych wierszy z poleceniami. Jeśli pierwszym wyrazem w danej linii nie jest żadne ze zdefiniowanych w dalszej części dokumentu poleceń (`dodaj`, `wypisz`, `usun`, `zapisz`, `zapiszwszystko`, `wczytaj`, `wczytajwszystko`, `koniec`) należy wypisać pojedynczy znak zapytania i zignorować tą linię. To samo dotyczy sytuacji gdy wprowadzona zostanie pusta linia.

Każda odpowiedź programu musi kończyć się pojedynczym znakiem końca linii. W wyjściu nie powinny pojawić się żadne puste linie.

Polecenie `koniec` oznacza żądanie zakończenia pracy programu bez przetwarzania żadnych kolejnych poleceń (i bez wypisywania komunikatu). Ponadto program powinien zakończyć się po napotkaniu znacznika końca pliku, który będzie przekazany automatycznie przy przekierowaniu strumienia wejściowego, lub wprowadzony ręcznie przy pracy interaktywnej (Windows: Ctrl-Z).

2.3. Dodawanie danych

Pierwszą funkcją jaką powinien obsługiwać projekt jest dodawanie danych (bez tego inne polecenia byłyby trochę bez sensu, prawda?).

format polecenia	`dodaj <nazwa tabeli>` `(<wartosc pierwszego atrybutu>,<wartosc 2>,…)`
przykłady	dodaj liczby (1) dodaj studenci (123456,”Jan”,”Kowalski”)

odpowiedź programu	`Liczba dodanych rekordow: 1`
	albo (przy pełnej tabeli)
	`Liczba dodanych rekordow: 0`

uwagi	Pomiędzy słowem dodaj a nazwą tabeli, oraz między nazwą tabeli a nawiasem znajduje się dokładnie jedna spacja. Jeśli tabela zawiera już 1000 rekordów należy zapobiec dodaniu kolejnego;

2.3.1. Dodawanie komórek w dowolnej kolejności

Polecenie dodaj zostaje rozszerzone o możliwość swobodnego wyboru atrybutów dodawanych rekordów. Dla przykładu: dodaj sale(rozmiar,nazwa) (10, "301") oznacza dodanie jednego rekordu do tabeli sale.

format polecenia	`dodaj <nazwa tabeli>` `(<wartosc pierwszego atrybutu>,<wartosc 2.>,…)`
przykłady	dodaj wyniki(indeks,srednia) (123456,0.234) dodaj studenci(imie,nazwisko) (”Jan”,”Kowalski”)

odpowiedź programu	`Liczba dodanych rekordow: 1`
	albo (przy pełnej tabeli)
	`Liczba dodanych rekordow: 0`

uwagi	nadal powinna działać podstawowa funkcjonalność polecenia dodaj (bez określenia listy atrybutów); między nazwą tabeli a nawiasem nie ma żadnego odstępu, podobnie wewnątrz nawiasów z listą atrybutów nie ma żadnych spacji); każdy atrybut pojawi się na liście co najwyżej raz; Jeśli tabela zawiera już 1000 rekordów należy zapobiec dodaniu kolejnego;

Uwaga: jeśli chcesz uzyskać punktację zarówno za dodawanie komórek w dowolnej kolejności, jak i autoinkrementację lub ograniczanie wartości – musisz je wpierać również w tym trybie dodawania.

2.3.2. Dodawanie wielu rekordów równocześnie

Polecenie dodaj zostaje rozszerzone o możliwość podania wielu rekordów naraz.

format polecenia	`dodaj <nazwa tabeli>`
	`(<wartosc pierwszego atrybutu>,<wartosc 2.>,…)`
	`[(<wartosc pierwszego atrybutu>,<wartosc 2.>,…)]`
	`[(<wartosc pierwszego atrybutu>,<wartosc 2.>,…)] […]`

przykłady	dodaj liczby (1) (2) (6) (-2) (10) dodaj studenci (123,”Jan”,”Kowalski”) (123456,”Jacek”,”Kowalski”)
odpowiedź programu	`Liczba dodanych rekordow: <liczbaDodanychRekordow>`
uwagi	Po nazwie tabeli może znajdować się dowolna liczba rekordów – każdy „opakowany” nawiasami, pomiędzy którymi jest dokładnie jedna spacja odstępu; Jeśli dodanie któregoś rekordu zwiększyłoby rozmiar tabeli ponad 1000 należy przerwać operację.

Uwaga: jeśli chcesz uzyskać punktację zarówno za obsługę wielu rekordów, jak i autoinkrementację lub ograniczanie wartości – musisz je wpierać również w trybie dodawania wielu rekordów naraz.

2.3.3. Autoinkrementacja

W przypadku autoinkrementacji należy dopuścić dwa sposoby dodawania wierszy do tabeli przedmioty – z trzema lub z dwiema wartościami atrybutów. W drugim przypadku należy założyć, ze są to nazwa oraz numer semestru:

```
dodaj przedmioty(*,"Programowanie",1)
```

W takim wypadku wartość atrybutu `id` ma być ustawiona na najmniejszą wartość dodatnią, która nie występuje wśród istniejących rekordów (zobacz przykład w rozdziale 3.1.2). W przypadku podania trzech atrybutów nie trzeba sprawdzać unikalności wartości – należy zaufać, że użytkownik wie co robi.

2.3.4. Ograniczenie zakresu atrybutu

W przypadku implementacji ograniczenia zakresu składnia polecenia dodaj pozostaje bez zmian, ale wartość wstawiana do pola `semestr` (tablica `studenci`) ma być ustawiona na 1, jeśli użytkownik podał mniejszą od 1 lub 10, jeśli podał większą od 10. W przypadku obsługi czterech tabeli należy odpowiednio obsłużyć dla atrybutu `rozmiar` w tabeli `sale`.

3. Wyświetlanie danych

Kolejnym istotnym z punktu widzenia użytkownika poleceniem jest polecenie służące do wyświetlenia zawartości tabel. W najprostszej postaci przyjmuje ono formę:

format polecenia	`wypisz <nazwa tabeli>`
przykłady	wypisz liczby wypisz studenci

odpowiedź programu	`Liczba rekordow: <liczba>`
	`<nazwa atrybutu 1> <nazwa2> …`
	`<wartosc1_1> <wartosc2_1> <wartosc3_1>`
	`<wartosc1_2> <wartosc2_2> <wartosc3_2>`

uwagi	Wyświetl zawartość tabeli posortowaną według kolejnych pól (zaczynając od lewej). Jeśli w odpowiedzi jest 0 rekordów to nazw atrybutów nie należy wyświetlać.

Podstawową formę wyświetlania zawartości należy zaimplementować dla wszystkich tabel.

3.1.1. Postać rozbudowana zapytania

Postać rozbudowana może być zaimplementowana jedynie dla tabel `liczby` oraz `studenci`. W swojej najbardziej rozbudowanej postaci polecenie będzie się składać z bloków w określonej kolejności:

```
wypisz <nazwa tabeli>[kolumny] [sortowanie] [wybór] [zakres]
```

gdzie kolejne opcjonalne bloki w nawiasach kwadratowych zawierają odpowiednio określenie kolumn do wypisania, sposób sortowania, kryterium wyboru i zakres rekordów do wyświetlenia.

Na przykład: `wypisz studenci(imie,nazwisko) wedlug nazwisko gdzie indeks>100 zakres 0-9`
oznacza polecenie wypisania imienia i nazwiska dziesięciu pierwszych ((studentów o numerze indeksu większym od 100) posortowanych alfabetycznie)[4].

3.1.2. Wybór kolumn

Bezpośrednio po nazwie tabeli (bez spacji) może pojawić się nawias, a w nim lista kolumn, które mają znaleźć się w odpowiedzi programu. Uwaga – użytkownik może życzyć sobie wielokrotnego wypisania każdej z kolumn).

Wejście programu	Wyjście programu

`dodaj studenci (124,”Jan”,”Now”)`
`(123,”Ada”,”Now”) (*,”Tom”,”Mor”)`
 	
`wypisz studenci(imie,id,id)`
 	
`wypisz studenci(nazwisko,imie)`
	
	`Liczba dodanych rekordow: 3`
	
	`Liczba rekordow: 3`
	`imie id id`
	`Tom 1 1`
	`Ada 123 123`
	`Jan 124 124`
	
	`Liczba rekordow: 3`
	`imie id id`
	`Mor Tom`
	`Now Ada`
	`Now Jan`

[4] Nawiasy jak w matematyce – najpierw wyznaczamy wartość wewnątrz, a dopiero potem stosujemy inne operacje.

3.1.3. Zakres wyświetlania

Opcja określenia zakresu ma pozwolić użytkownikowi na wyświetlenie tylko części rekordów – przy czym kryterium wyboru nie zależy bezpośrednio od danych – a po prostu od kolejności na liście odpowiedzi (kolejne strony jednej tabeli).

Jeśli po określeniu tabeli (i ew. kolumn) pojawi się słowo `zakres` to bezpośrednio po nim znajdzie się spacja, a po niej dwie liczby całkowite oddzielone myślnikiem: *min* oraz *max*. Zakładając, że w odpowiedzi na zapytanie mamy *x* rekordów (indeksowanych od 0 do *x*-1) limitowanie ma działać następująco:

- nie są wyświetlane wiersze odpowiedzi o indeksach mniejszych od *min*,
- nie są wyświetlane wiersze odpowiedzi o indeksach większych od *max*,
- wiersz „`Liczba rekordow: <liczba>`” ma zawierać liczbę rekordów spełniających powyższe warunki.

Wejście programu	Wyjście programu

`dodaj liczby (1) (2) (3) (4) (5) (6) (7)`
 	
`dodaj liczby (8) (9) (10)`
 	
`wypisz liczby zakres 0-1`
 	
`wypisz liczby gdzie wartosc>4 zakres 1-3`
	
	`Liczba dodanych rekordow: 7`
	
	`Liczba dodanych rekordow: 3`
	
	`Liczba rekordow: 2`
	`wartosc`
	`1`
	`2`
	
	`Liczba rekordow: 3`
	`wartosc`
	`6`
	`7`
	`8`

3.1.4. Kryterium sortowania

Jeśli w zapytaniu pojawi się słowo `wedlug` to bezpośrednio za nim znajdzie się spacja, a następnie nazwa atrybutu lub wykrzyknik i nazwa atrybutu (bez dodatkowej spacji), według którego mają być posortowane odpowiedzi. Kolejnymi kryteriami sortowania są atrybuty zgodnie z definicją tabeli, począwszy od pierwszego. Jeśli nazwa atrybutu poprzedzona jest wykrzyknikiem, to wynik sortowania należy odwrócić.

Wejście programu	Wyjście programu

`dodaj liczby (1) (4) (3) (2)`
 	
`wypisz liczby wedlug !wartosc`
 	
`wypisz liczby wedlug !wartosc`
`gdzie wartosc>2 limit 1-1`
	
	`Liczba dodanych rekordow: 4`
	
	`Liczba rekordow: 4`
	`wartosc`
	`4`
	`3`
	`2`
	`1`
	
	`Liczba rekordow: 1`
	`wartosc`
	`3`

Uwaga: Pomiędzy `!wartosc` a `gdzie` jest dokładnie jedna spacja.

3.1.5. Kryterium wyboru – wersja uproszczona

Wybór pozwala określić, które rekordy zostaną wypisane na podstawie danych w nich zawartych. Po słowie `gdzie` znajduje się spacja, a następnie warunek postaci `<nazwaAtrybutu><operator><wartosc>`. Operatory to <, >, =, ! czyli kolejno: mniejsze od, większe od, równe i różne od (sam wykrzyknik).

W podstawowej wersji wystarczy by określenie warunku poprawnie działało dla atrybutu `indeks` w tabeli `studenci`.

3.1.6. Kryterium wyboru – wersja pełna

Użytkownik ma możliwość swobodnego zdefiniowania warunku w obrębie tabeli studenci. Działanie operatorów mniejsze od oraz większe od dla typów `bool` i `string` są opisane w rozdziale 2.1.1 (w skrócie: `bool`: nie obsługiwać bo nie będą wprowadzane, `string`: alfabetycznie, z uwzględnieniem wielkości liter).

3.2. Usuwanie danych

Jak łatwo się domyślić, w tej części zdefiniowane będzie polecenie służące do usuwania danych. W prostszym wariancie usuwanie ma działać wyłącznie dla określonych atrybutów w dwóch tabelach, w trudniejszym – dla dowolnych atrybutów.

format polecenia	`usun <nazwa tabeli>` `usun <nazwa tabeli> gdzie <warunek>`
przykłady	usun liczby usun liczby gdzie wartosc=4 usun studenci gdzie nazwisko>”N”
odpowiedź programu	`Liczba usuniętych rekordow: <liczba>`
uwagi	Definicja warunku jest identyczna jak w przypadku wyświetlania zawartości tablicy (rozdział 3.1.5).

3.2.1. Podstawowe kryterium wyboru

Jeśli w poleceniu `usun` znajduje się słowo „`gdzie`”, to bezpośrednio po nim znajduje się spacja, a następnie warunek postaci `<nazwaAtrybutu><operator><wartosc>`. Operatory to <, >, =, ! czyli kolejno: mniejsze od, większe od, równe i różne od (sam wykrzyknik).

Wymagane jest, aby polecenie postaci `usun … gdzie …` działało dla wszystkich tabel dla pierwszego atrybutu. Jeśli więcej niż jeden element spełnia warunek – usunięte mają zostać wszystkie z nich.

3.2.2. Rozszerzone kryterium wyboru

Aby zrealizować tą funkcjonalność należy sprawić, że w kryterium usuwania możemy określić dowolny warunek (dla każdego atrybutu z każdej tabeli).

3.3. Zapis i odczyt

Zapis i odczyt danych można zrealizować w dwóch wariantach: w pierwszym operacje dotyczą wszystkich tabel do jakichś predefiniowanych plików (lub też jednego); w drugim natomiast to użytkownik określa jakiej tabeli ma dotyczyć operacja oraz podaje nazwę pliku. Otrzymanie pełnej punktacji wymaga implementacji obu mechanizmów.

3.3.1. Wszystko naraz

Program powinien obsługiwać dwa polecenia:

format polecenia	`zapiszwszystko`
odpowiedź programu	`OK` (nawet jeśli wystąpi błąd)
uwagi	w bieżącym katalogu tworzy plik(i) zawierające wszystkie informacje aktualnie znajdujące się w tabelach; program nie wyświetla żadnego komunikatu.

format polecenia	`wczytajwszystko`
odpowiedź programu	`OK` (nawet jeśli wystąpi błąd)
uwagi	z bieżącego katalogu odczytuje wszystkie dane z pliku(ów); jeśli przy odczycie któregokolwiek pliku wystąpi błąd to wszystkie tabele mają zostać wyczyszczone; program nie wyświetla żadnego komunikatu.

3.3.2. Obsługa pojedynczych tabel

Program powinien obsługiwać dwa polecenia:

format polecenia	`zapisz <nazwa tabeli> <nazwa pliku>`
przykład	zapisz studenci absolwenci.dat
odpowiedź programu	`OK` (nawet jeśli wystąpi błąd)
uwagi	w bieżącym katalogu tworzy plik zawierający identyfikator typu tabeli oraz wszystkie dane znajdujące się w niej; można przyjąć założenie, że nazwa pliku nie zawiera spacji; program nie wyświetla żadnego komunikatu.

format polecenia	`wczytaj <nazwa tabeli> <nazwa pliku>`
przykład	wczytaj studenci absolwenci.dat
odpowiedź programu	`OK` (nawet jeśli wystąpi błąd)
uwagi	z bieżącego katalogu odczytuje wskazany plik, i jeśli zgadza się identyfikator tabeli to umieszcza dane w nim zawarte w pamięci; jeśli przy odczycie któregokolwiek pliku wystąpi błąd to wskazana tabela ma zostać wyczyszczona program nie wyświetla żadnego komunikatu.

4. Podsumowanie

Zdajemy sobie sprawę, że dla osób zaczynających programowanie opis projektu może wydawać się nieco przerażający. Z jednej strony trochę taki ma być (trzeba się wziąć do roboty), z drugiej – proszę obciąć funkcjonalność programu do rzeczy wymaganych:

- obsługa int i string, trzy tabele,
- dodawanie całych wierszy,
- wyświetlanie całej zawartości posortowanej po kolejnych polach wszystkich tablic,
- Skasuj - pojedynczy warunek we wszystkich tabelach,
- Zapis i odczyt wszystkiego co jest we wszystkich tabelach

Od tego należałoby zacząć, a potem spróbować implementacji kolejnych rozszerzeń.