YAC Data Builder

wersja 4.13

Podręcznik użytkownika

Spis treści

Rozdział

1 Wprowadzenie

Program YAC Data Builder służy do przygotowywania badań do analizy pod aplikacją YAC Data Analyzer.

1.1 Wymagania

Program YAC Data Builder działa wyłącznie na komputerach PC pracujących pod systemami operacyjnymi Microsoft: Windows 9x, Windows NT, Windows 2000, Windows XP lub ich nowszymi wersjami.

1.2 Instalacja

Programu YAC Data Builder nie trzeba w żaden specjalny sposób instalować - wystarczy wgrać go do dowolnego katalogu na dysk komputera. Prosimy jednak pamiętać, że do przetwarzania badań chronionych niezbędny jest dostęp aplikacji do bazy danych licencji umieszczonej na serwerze firmy YAC Software.

1.3 Ochrona

Program YAC Data Builder jest chroniony przed nielegalnym kopiowaniem - wersja pełna programu wymaga dostępu do serwera licencji firmy YAC Software. Szczegóły dostępu do serwera licencji podawane są przy zakupie programu.

Uwaga
Powyższe nie dotyczy wersji Lite.

1.4 Wersja Lite

Wersja Lite programu YAC Data Builder jest udostępniana nieodpłatnie wszystkim zainteresowanym.

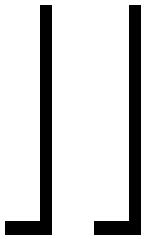
Pozwala na przetwarzanie danych o wielkości do 1100 przypadków i 100 kolumn (oba warunki muszą być spełnione). Nie pozwala na przetwarzanie badań chronionych ani badań zawierających specjalistyczne analizy czytelnictwa prasy i słuchalności radia.

1.5 Kontakt i pomoc techniczna

W razie pytań dotyczących programów i procedur wymienionych w niniejszym dokumencie prosimy o kontakt:

YAC Software
support@yac.com.pl
www.yac.com.pl

Rozdział



2 Działanie

Program YAC Data Builder przetwarza pliki wejściowe na format wykorzystywany przez program YAC Data Analyzer (pliki wyjściowe).

Takie przetwarzanie niezbędne jest z dwóch względów:
- zoptymalizowania czasu dostępu do danych i zmniejszenia rozmiaru samych danych,
- zaszyfrowania danych i opisów tak, aby użytkownik końcowy nie mógł na ich podstawie odtworzyć danych źródłowych.

Ponadto, w trakcie przetwarzania danych, program YAC Data Builder sprawdza poprawność opisów danych oraz spójność samych danych.

2.1 Pliki wejściowe

Do przetworzenie badania wymagane są następujące pliki wejściowe:
- plik opisowy badania o rozszerzeniu `.dbs`,
- pliki z danymi (w podziale na poszczególne fale) oraz pliki opisów lokalizacji kolumn w danych.

Ponadto dane wejściowe badania mogą być uzupełnione o następujące pliki:
- ikona firmy / badania,
- logo firmy / badania,
- pliki informacyjne.

2.1.1 Plik opisowy badania

Badanie, które chcemy udostępniać klientom za pomocą programu YAC Data Builder musi zostać opisane w pliku tekstowym o rozszerzeniu `.dbs`.

Plik ten musi być zapisany w czystym ASCII, bez jakichkolwiek znaków formatujących. Oznacza to, że edytując plik np. w programie MS Word, należy go zapisać jako plik tekstowy, a nie jako dokument tej aplikacji.

W pliku tym znajdą się opisy m.in. fal, wag, pytań, odpowiedzi i wielu innych parametrów. Opisane są także pliki, w których trzymane są dane i lokalizacje kolumn (instrukcja `recordset`).

Dokładny opis języka znajduje się w rozdziale Język dokumentacji badań.

2.1.2 Pliki danych i lokalizacji kolumn

Poprawnie przygotowane do przetworzenia zbiory danych powinny być zapisane w formacie Fixed-ASCII, tzn. odpowiednie zmienne w każdym rekordzie powinny znajdować się w tych samych kolumnach, a każdy rekord powinien zajmować jedną linię i być tej samej długości (składać się z takiej samej liczby znaków).

W jednym takim pliku może znajdować się tylko jedna fala badania.

Ponieważ na podstawie samego pliku danych nie wiadomo, w których kolumnach znajdują się odpowiedzi na różne pytania, potrzebne są dodatkowe pliki opisujące te kolumny. Zatem, do każdego pliku danych potrzebny jest opis jego struktury. Jeżeli dwie lub więcej kolejnych fal mają tę samą strukturę kolumn, wystarczy opisać pierwszą z nich.

Dostępne są dwa formaty plików opisujących zbiory danych obsługiwane przez YAC Data Builder:

Plik w formacie SPSS, np.:

```
DATA LIST FILE='FALA1.DAT'/
V1 1
V2 2-4
V3 5-24 (A)
V4 25
...
```

gdzie `Vk` oznacza nazwy zmiennych. Informacje o formacie zmiennych, etykietach zmiennych i wartości, brakach danych (missing values) są przez program YAC Data Builder ignorowane.

Plik w formacie prostym, np.:

```
V1 L1
V2 L3
V3 L20
V4 L1
...
```

gdzie `Vk` oznacza nazwy zmiennych a `Ln` oznacza liczbę zajmowanych przez daną zmienną kolumn.

Wszystkie pliki danych i lokalizacji kolumn powinny znajdować się w tym samym katalogu co plik `.dbs`.

Do uzyskania plików w powyższych formatach z danych w formacie programu SPSS (`.sav`) służy program YAC Data Converter.

Patrz też instrukcja `recordset`.

2.1.3 Pliki graficzne

Do dystrybucji można dodać plik z ikoną specyficzną dla danego badania. Ikona ta będzie wyświetlana w programie YAC Data Analyzer po otwarciu danych. Ikona jest także widoczna w Eksploratorze Windows.

Plik powinien mieć rozszerzenie `.ico` i być zapisany w formacie ikony stosowanym w systemie Windows.

Patrz też instrukcja `icon`.

Do dystrybucji można także dodać plik z dowolną bitmapą (np. logo firmy badawczej). Bitmapa będzie wyświetlana w pasku stanu po otwarciu badania.

Plik powinien mieć rozszerzenie `.bmp` i być zapisany w formacie bitmapy stosowanym w systemie Windows.

Patrz też instrukcje `logo` i `logoPlacement`.

2.1.4 Pliki informacyjne

Do dystrybuowanych danych można opcjonalnie dołączyć dowolne pliki informacyjne w dowolnym formacie (rozpoznawalnym przez system Windows). Mogą więc być to pliki MS Office (Word, Excel, PowerPoint), pliki PDF czy też pliki stron internetowych HTML.

Po wybraniu odpowiedniej opcji, w Menedżerze Badań programu YAC Data Analyzer zostaną wyświetlone pliki informacyjne.

Patrz też instrukcja `infopage`.

2.2 Licencje

Badania mogą być chronione przed nielegalną dystrybucją za pomocą systemu licencji firmy YAC Software.

2.2.1 Badania darmowe i chronione

Program YAC Data Builder może przetworzyć badania na dwa sposoby:
- jako badanie darmowe,
- jako badanie chronione.

Badanie darmowe to badanie, które nie jest zabezpieczone przed nielegalną dystrybucją. Każda osoba, która będzie miała dostęp do pliku `.das` badania, będzie mogą prowadzić analizy na tych danych w programie YAC Data Analyzer.

Badania mogą być chronione przed nielegalną dystrybucją - aby takie badanie uruchomić na komputerze użytkownika, komputer musi być zarejestrowany dla tego badania w bazie danych systemu YAC Licensing Kit (dostępnego za oddzielną opłatą). Program YAC Data Builder przetwarzając dane łączy się z bazą danych licencji i następnie kody komputerów przypisanych do przetwarzanego badania dodawane są do pliku wyjściowego `.das`. Przetworzone badanie może być wtedy uruchamiane tylko na tych zarejestrowanych komputerach.

Do rejestracji komputerów służą dwa programy:
- YAC Code Generator (`YCG.exe`) do pobierania kodów komputerów od użytkowników,
- YAC License Manager (`YLM.exe`) do wpisywania kodów tych komputerów do bazy danych licencji.

Jeżeli zostaną dodane nowe komputery do tego badania, badanie należy przetworzyć kolejny raz, aby kody nowych komputerów zostały dodane do wynikowego pliku `.das`.

Patrz także instrukcja `licenses`.

Uwaga
Ochrona badań nie jest dostępna w wersji Lite.

2.3 Pliki wynikowe

W wyniku działania programu powstanie plik `.das`, w którym znajdują się:
- przetworzony plik `.dbs`,
- przetworzone pliki danych,
- przetworzone pliki lokalizacji kolumn.

2.3.1 Błędy, ostrzeżenia, uwagi

Jeżeli w dokumentacji badania, plikach danych lub plikach lokalizacji pojawią się błędy, to:
- program kończąc działanie zgłosi, że nie wszystko jest w porządku i poda przyczynę problemów,
- błędy dotyczące dokumentacji badania zostaną umieszczone, jako komentarze, w pliku `.dbs`,
- plik `.das` nie powstanie i / lub zostanie usunięty.

Drugi punkt może wymagać wyjaśnień:
- Plik `.dbs` jest plikiem tekstowym, w którym tekst po znakach `//` jest przez program YAC Data Builder ignorowany - są to komentarze, służące do opisania samego pliku `.dbs`.
- Jeżeli program YAC Data Builder znajdzie błędy w tym pliku (np. brak zamykającego **`end`** do definicji **`def`**), wtedy taka informacja zostanie umieszczona w pliku `.dbs` po znakach `//!` (więc jako komentarz). Wtedy użytkownik w łatwy sposób, za pomocą zwykłego wyszukiwania, może znaleźć wszystkie problematyczne miejsca.
- Program YAC Data Builder przy przetwarzaniu dokumentacji badania, najpierw usuwa wszystkie komentarze z pliku `.dbs`, które zaczynają się znakami `//!` (więc informacje o błędach z poprzedniego uruchomienia są usuwane aby nie wchodziły w konflikt z nowymi komunikatami). Oznacza to też, że samemu nie należy rozpoczynać komentarzy znakami `//!`, gdyż przy następnym przetwarzaniu zostaną one usunięte.

2.3.2 Dystrybucja

Plik wynikowy `.das`, który powstaje w wyniku działania programu YAC Data Builder na poprawnych i poprawnie opisanych danych, jest plikiem gotowym do uruchomienia pod aplikacją YAC Data Analyzer. Zawiera on w sobie przetworzone zbiory danych i ich specyfikacje, przetworzony plik opisowy badania `.dbs` oraz, w przypadku badania chronionego, informacje o licencjach przypisanych do danego badania.

Plik `.das` nie zawiera ikony / logo badania (musi być ona dostępna w systemie Windows) ani plików informacyjnych (muszą być dostępne jako oddzielne pliki w systemie Windows). Wysyłając zatem badania do klienta, należy wysyłać następujące pliki:
- plik `.das`,
- pliki ikony / logo (jeżeli są),
- pliki informacyjne (jeżeli są).

Pliki badania nie wymagają u klienta specjalnej instalacji - wystarczy je skopiować na dysk komputera.

2.4 Wersja konsolowa

Z programem `YDB.exe` (działającym jako aplikacja GUI) dystrybuowana jest jego wersja konsolowa - program `YDBC.exe`. Pozwala ona na automatyczne przetwarzanie badania po podaniu odpowiednich parametrów przy uruchomieniu programu.

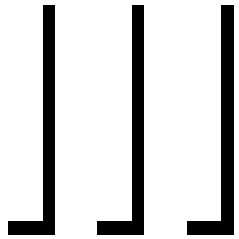
Program `YDBC.exe` najwygodniej uruchamiać w oknie konsoli systemu Windows (vel Tryb MS-DOS).

Parametry programu to:
- opcjonalny parametr `-d` oznaczający, że program nie powinien przetwarzać danych, tylko sprawdzić poprawność syntaktyczną dokumentacji badania (może to być przydatne wtedy, gdy dane mają dużą objętość i ich wielokrotne przetwarzanie zajmuje zbyt dużo czasu); jak w wersji graficznej, błędy zostaną dodane do dokumentacji jako komentarze rozpoczynające się znakami `//!`
- obowiązkowy parametr - nazwa pliku `.dbs` (łącznie z rozszerzeniem).

Po podaniu tych parametrów program zacznie przetwarzać dokumentację badania i ewentualnie dane.

Przykłady uruchomienia:
```
YDBC.exe "C:\Moje Badania\Badanie1\Badanie1.dbs"
YDBC.exe -d Badanie2.dbs
```

Rozdział



3 Skróty klawiszowe

Wiele operacji w programie można wykonać (i często przyspieszyć) korzystając z następujących skrótów klawiszowych:

Nawigacja po pliku

przejście o linię w górę / w dół	strzałka w górę / w dół
przejście o jeden znak w lewo / w prawo	strzałka w lewo / w prawo
przesunięcie pliku o jedną linię w górę / w dół	`Ctrl` + strzałka w górę / strzałka w dół
przejście o słowo w lewo / w prawo	`Ctrl` + strzałka w lewo / w prawo
przejście o stronę w górę / w dół	`PgUp` / `PgDn`
przejście na górę strony / na dół strony	`Ctrl+PgUp` / `PgDn`
przejście na początek / na koniec linii	`Home` / `End`
przejście na początek / na koniec pliku	`Ctrl+Home` / `End`

Uwaga: jednoczesne wciśnięcie klawisza Shift powyższymi klawiszami powoduje, że w trakcie przesuwania migawki pozycjowskaźniej program będzie zaznaczał blok; blok ten następnie można np. usunąć, skopiować do schowka czy przesunąć.

Edycja podstawowa

zmiana trybu między wstawianiem a nadpisywaniem	`Ins`
kopiowanie do schowka	`Ctrl+C` lub `Ctrl+Ins`
wycinanie do schowka	`Ctrl+X` lub `Shift+Del`
wstawianie ze schowka	`Ctrl+V` lub `Shift+Ins`
usuwanie bloku	`Ctrl+Del`
usuwanie znaku za migawką	`Del`
usuwanie znaku przed migawką	`Backspace`
usuwanie słowa za migawką	`Ctrl+T`
usuwanie słowa przed migawką	`Ctrl+Backspace`
cofanie ostatnich operacji (Undo)	`Ctrl+Z` lub `Alt+Backspace`
ponawianie cofniętych operacji (Redo)	`Shift+Ctrl+Z` lub `Shift+Alt+Backspace`
zaznaczenie całego pliku	`Ctrl+A`
przesunięcie zaznaczonego bloku w prawo	`Shift+Ctrl+I`
przesunięcie zaznaczonego bloku w lewo	`Shift+Ctrl+U`

Edycja podstawowa, c.d.

usunięcie linii	`Ctrl+Y`
usunięcie do końca linii	`Shift+Ctrl+Y`
standardowe zaznaczanie	`Shift+Ctrl+N`
zaznaczanie kolumn	`Shift+Ctrl+C`
zaznaczanie linii	`Shift+Ctrl+L`
przejdź do poprzedniego okna	`Shift+Ctrl+Tab`
przejdź do następnego okna	`Ctrl+Tab`
lista otwartych okien	`Alt+0` (cyfra zero)
zamknij okno	`Ctrl+F4`
zakończ pracę z programem	`Alt+F4`

Funkcje szukania i zamiany

znajdź	`Ctrl+F`
znajdź i zamień	`Ctrl+H`
powtórz ostatnią operację szukania / zamiany	`F3`
szukanie przyrostowe	`Ctrl+F3`
przejdź do danej linii	`Ctrl+L`
wstawienie zakładki n (gdzie n jest cyfrą)	`Shift+Ctrl+n` (np. `Shift+Ctrl+2`)
przejście do zakładki n	`Ctrl+n` (np. `Ctrl+5`)

Obsługa makr i przetwarzania

rozpocznij / zakończ rejestrację makra	`Shift+Ctrl+R`
uruchom makro	`Shift+Ctrl+P`
sprawdź dokumentację	`Ctrl+F9`
przetwarzaj dane	`F9`
uruchom program YAC Data Analyzer z przetworzonym badaniem	`F10`
przejdź do poprzedniego błędu	`F7`
przejdź do następnego błędu	`F8`

Operacje plikowe

nowy plik	`Ctrl+N`
importuj plik	`Ctrl+I`
otwórz plik	`Ctrl+O` (litera o)
zapisz plik	`Ctrl+S`

Rozdział IV

4 Język dokumentacji badań

Poniższy temat opisuje wszelkie kwestie związane z przygotowaniem dokumentacji badania - pliku `.dbs`.

4.1 Notacja

W niniejszym rozdziale używane są następujące czcionki do wyróżnienia niektórych elementów. I tak:

- `zawartość i nazwy plików, jak również przykłady dokumentacji są zapisane czcionką Courier New`,
- **słowa zarezerwowane języka są pogrubione**,
- *komentarze w dokumentacji są zapisane tekstem pochyłym*,
- *literały tekstowe są także zapisane tekstem pochyłym*.

4.2 Gramatyka

W opisach struktury instrukcji korzystamy z następującego sposobu zapisu:

- tekst, który należy zapisać dosłownie (bez dodatkowych znaków, np. spacji) jest ujmowany w cudzysłów,
- elementy opcjonalne umieszczane są w nawiasach kwadratowych (samych nawiasów nie należy wpisywać),
- elementy, które mogą się powtarzać wielokrotnie (ale mogą się nie pojawić w ogóle), są umieszczane w nawiasach klamrowych (samych nawiasów nie należy wpisywać),
- inne elementy są opisywane słownie (jako jeden ciąg znaków, np. `nazwa_pliku`); do tych elementów dopisywana jest informacja, jakiej postaci mogą być to elementy (np. tekst, identyfikator, etc.).

Przykłady:

Opis instrukcji
```
"file" "=" nazwa_pliku ";"
```
gdzie nazwa pliku jest literałem tekstowym, opisuje następującą instrukcję:
```
file = "c:\Doc.txt";
```

Natomiast
```
"file" "=" nazwa_pliku [ "," nazwa_pliku ] ";"
```
opisuje następujące instrukcje:
```
file = "c:\Doc.txt";
file = "c:\Doc.txt", "c:\Info.txt";
```
ale już nie:
```
file = "c:\Doc.txt", "c:\Info.txt", "c:\News.txt";
```

Ostatni przykład zgodny jest natomiast z opisem następującym:
```
"file" "=" nazwa_pliku { "," nazwa_pliku } ";"
```

4.3 Informacje ogólne

Zanim omówimy dokładnie poszczególne elementy opisu badań, najpierw dokładniej opiszemy elementy, jakie w takiej dokumentacji mogą się pojawić.

4.3.1 Formatowanie

Dokumentacja powinna zostać zapisana w pliku tekstowym bez dodatkowych instrukcji formatujących. Pisząc dokumentację np. w program MS Word, należy plik zapisać jako plik tekstowy.

W pliku mogą pojawiać się spacje i nowe linie wszędzie tam, gdzie nie dzielą one podstawowych elementów dokumentacji (jak identyfikatory, liczby czy teksty).

4.3.2 Komentarze

Komentarze zapisane w dokumentacji badania są przez program ignorowane - służą do zapisania dodatkowych uwag, które np. tłumaczą, dlaczego przyjęto takie rozwiązania, a nie inne.

Przyjęto oznaczenia na komentarze takie, jak w językach programowania C czy Java, czyli:
- komentarze do końca linii zaczynają się dwoma znakami slash `//`,
- komentarze wielolinijkowe zaczynają się od `/*` i kończą na `*/`; komentarze te nie mogą być zagnieżdżane (komentarz w komentarzu).

Przykład:

```
def question
  id = sex;
  name = "Płeć";
  // Należy skorzystać z kolumny SEX a nie M1 -
  // braki danych zostały uzupełnione na podstawie kart doboru.
  column = SEX;
  def responseList
    def response id = male;   code = 1; name = "mężczyźni"; end;
    def response id = female; code = 2; name = "kobiety"; end;
  end;
end;
```

W powyższym przykładzie zdefiniowano opis pytania o płeć respondenta. Tekst komentarza (dwie linie zaczynające się podwójnymi znakami łamania) jest przez program YAC Data Builder ignorowany.

4.3.3 Identyfikatory

Identyfikatory służą do identyfikacji wszelkich elementów definiowanych w pliku opisowym badania, jak np. wag, fal, zbiorów danych, pytań, odpowiedzi, itp.

Identyfikatory nadawane są przez autora pliku i powinny możliwie dokładnie odzwierciedlać znaczenie definiowanego obiektu. Definiując zatem np. identyfikator pytania, lepiej posłużyć się nazwą typu `WSP` (od znajomość wspomaganej) niż np. `P1`.

Niezależnie, czy autor zdecyduje się na mniej czy bardziej znaczące identyfikatory, ich postać ograniczona jest kilkoma regułami:
- identyfikatory muszą się zaczynać literą,
- kolejne znaki mogą być literami, cyframi, znakiem podkreślenia lub kropką,
- nie są dozwolone litery ze znakami diakrytycznymi.

Dopuszczalne są zatem identyfikatory: `wiek`, `user.heavy`, `prasa_handlowa` czy `fala1`. Niedopuszczalne są np. następujące identyfikatory:
- `płeć` (nie można korzystać z polskich liter),
- `user-heavy` (myślnik jest niedopuszczalny),
- `prasa handlowa` (identyfikator musi być jednym ciągiem znaków bez spacji),
- `2fala` (na początku musi być litera).

W nazwach identyfikatorów nie są rozróżniane wielkie i małe litery. Definiując element o identyfikatorze sex można się później do niego odwoływać pisząc `Sex`. Sugerujemy jednakże, aby trzymać się jednolitego sposobu zapisu identyfikatorów, gdyż zwiększa to czytelność pliku opisowego badania.

4.3.4 Teksty

Elementy opisywane w pliku badania będą wyświetlane następnie w programie YAC Data Analyzer. Należy zatem zdefiniować opisy słowne tych elementów.

W pliku opisowym badania mogą pojawiać się teksty jednolinijkowe (jak np. nazwa pytania) lub wielolinijkowe (np. treść pytania).

Teksty jednolinijkowe umieszcza się między cudzysłowami. Jeżeli w samym tekście chcemy umieścić cudzysłów, należy go zapisać dwukrotnie:

- *"Poprawny napis bez cudzysłowu"*,
- *"Błędny napis z " cudzysłowem"* (cudzysłów wewnątrz tekstu musi zostać powielony, jak niżej),
- *"Poprawny napis z "" cudzysłowem"* (zostanie wyświetlony jako: *Poprawny napis z " cudzysłowem*).

Teksty wielolinijkowe rozpoczyna się dwoma znakami mniejszości << a kończy dwoma znakami większości >>.

Uwaga
Tylko w niektórych miejscach dozwolone są teksty wielolinijkowe - w dalszej części niniejszej dokumentacji będą one dodatkowo opisane. Jednak wszędzie gdzie są one dopuszczalne, można podać tekst jednolinijkowy.

Uwaga
W tekstach, w przeciwieństwie do identyfikatorów, rozróżniane są wielkie i małe litery.

4.3.5 Wartości logiczne

W niektórych miejscach dokumentacji należy podać wartość logiczną. Należy wtedy wpisać 0 (zero) jako fałsz lub 1 (jeden) jako prawda.

Przykład:

```
def survey
  demo = 1;
end;
```

W powyższej definicji badania podano, że dane dystrybuowane są w wersji demonstracyjnej.

4.3.6 Liczby

W niektórych definicjach podajemy liczby - np. kody odpowiedzi.

Tam gdzie można podać liczbę rzeczywistą, należy ją zapisać z kropką jako separatorem części dziesiętnej - zapis ten jest niezależny od ustawień regionalnych komputera.

Natomiast w tekstach, liczby rzeczywiste można zapisywać dowolnie, lecz raczej należy zachować zgodność ze standardem regionalnym dla wersji językowej, w jakiej tekst jest zapisywany (patrz dokumentacja języków badań w definicji environ). Np. gramaturę czy ceny w polskiej wersji tekstów lepiej zapisywać z przecinkiem, a w angielskiej - z kropką.

4.3.7 Definicje

Dokumentacja badania składa się z definicji poszczególnych elementów (modułów, pytań, odpowiedzi, fal, wag, zbiorów danych, itp.).

Definicje te dzieli się na dwa typy:

- proste,
- złożone.

4.3.7.1 Definicje proste

Definicje proste służą do określania pojedynczych elementów lub list podobnych elementów.

Składnia ogólna:

```
nazwa_elementu "=" wartość { "," wartość } ";"
```

Przykłady:

```
demo = 1;
name = "Badanie demonstracyjne";
wizards = general, press;
```

Uwaga
Została tu opisana składnia ogólna definicji prostej, jednak większość takich definicji dopuszcza tylko jedną wartość, a nie listę.

4.3.7.2 Definicje złożone

Definicje złożone służą do wprowadzania bardziej skomplikowanych elementów do dokumentacji:

```
"def" nazwa_elementu
  definicja_elementu
"end" ";"
```

Gdzie `definicja_elementu` może składać się z definicji prostych i / lub złożonych.

Przykład:

```
def question
  id = sex;
  name = "Płeć";
  column = v1;
  def responseList
    def response
      id = male;
      code = 1;
      name = "mężczyźni";
    end;
    def response
      id = female;
      code = 2;
      name = "kobiety";
    end;
  end;
end;
```

W powyższym zapisie wcięcia służą do zwiększenia czytelności dokumentacji. Na ogół przyjmuje się, że **`def`** "widzi się" z odpowiadającym mu **`end`** (tzn. zaczynają się w tej samej kolumnie, a elementy pomiędzy zaczynają się na kolumnach dalszych). Powyższą definicję można jednak zapisać jak następuje:

```
def question
  id = sex;
  name = "Płeć";
  column = v1;
  def responseList
    def response id = male;   code = 1; name = "mężczyźni"; end;
    def response id = female; code = 2; name = "kobiety"; end;
  end;
end;
```

Sposób zapisu (czy bardziej rozwinięty, czy bardziej zwięzły) zależy tylko od upodobań osoby piszącej dokumentację - dla programu oba powyższe przykłady są nierozróżnialne.

4.4 Definicje wspólne

W definicjach wielu elementów definiuje się te same pola. W niniejszym punkcie omówione są definicje tych pól.

- `id` (identyfikator elementu - często definicja obowiązkowa)

  ```
  "id" "=" identyfikator ";"
  ```

  Identyfikator jest używany przez program YAC Data Analyzer przy zapisywaniu i późniejszym czytaniu raportów i definicji parametrów.

- `name` (nazwa elementu - definicja obowiązkowa)

  ```
  "name" "=" tekst_jednolinijkowy ";"
  ```

  Nazwa elementu wyświetlana jest przez program YAC Data Analyzer w dialogach i analizach.

- `nick` (skrócona nazwa elementu - definicja opcjonalna)

  ```
  "nick" "=" tekst_jednolinijkowy ";"
  ```

  W tabelach i na wykresach wyświetlenie pełnej nazwy elementu może nie być możliwe (ze względu na szerokość kolumny / miejsca zajmowane przez etykiety na wykresie). Wtedy program YAC Data Analyzer wykorzysta nazwę skróconą. Gdy element nick nie zostanie zdefiniowany, będzie wykorzystana nazwa (`name`) elementu.

- `info` (dodatkowa informacja o elemencie)

  ```
  "info" "=" tekst_wielolinijkowy ";"
  ```

  Dodatkowa informacja o elemencie wyświetlana przez program YAC Data Analyzer na żądanie użytkownika (może to być np. pełna treść pytania z kwestionariusza).

W dalszej części niniejszej dokumentacji, w definicjach gdzie występują wszystkie powyższe pola, będziemy pisać, że mogą występować pola standardowe.

Przykładem niech będzie część definicji pytania:

```
id = sex;
name = "Płeć";
info = "Pytanie nie zadawane - uzupełnione na podstawie kart doboru";
```

lub

```
id = spo;
name = "Spontaniczna znajomość marek";
nick = "SPO";
info = <<
  Respondentowi było zadawane pytanie o znajomość marek.
  Ankieter notował trzy pierwsze odpowiedzi.
>>;
```

4.5 Environ (środowisko)

Definicja ta służy określeniu podstawowych informacji o opisywanym badaniu.

Obecnie należy tu zdefiniować języki, w których dostępne będą opisy elementów badania (jak pytania, odpowiedzi, itp.)

```
"languages" "=" język { "," język } ";"
```

Przykład 1:

```
def environ
  languages = plk;
end;
```

Przykład 2:

```
def environ
  languages = plk, enu;
end;
```

Pierwsza definicja oznacza, że badanie jest dostępne tylko w języku polskim. Druga zaś mówi, że badanie jest dostępne zarówno po polsku jak i po angielsku (w wersji USA).

Powyższe identyfikatory języków muszą być zgodne z identyfikatorami przyjętymi w systemie Windows. Pełny spis języków zawarty został w aneksie A.

4.5.1 Korzystanie z języków w dalszej części dokumentacji

Po zdefiniowaniu języków badania, w dalszej części dokumentacji można poszczególne teksty tak opisywać, aby były one dostępne w każdym z wymienionych języków.

Jeżeli definiujemy np. nazwę elementu:

```
name = "Badanie demonstracyjne";
```

to w takiej postaci tekst "Badanie demonstracyjne" pojawi się w każdej wersji językowej badania (przełączając się na każdy z języków w aplikacji YAC Data Analyzer powyższy tekst pokazywany będzie w takiej samej postaci).

Załóżmy teraz, że chcielibyśmy, aby w wersji angielskiej pojawił się tekst "Demonstration survey". Służą do tego znaczniki języków, jak pokazuje przykład poniżej:

```
name = "<enu>Demonstration survey<plk>Badanie demonstracyjne";
```

Od tekstu `<enu>` do następnego znacznika definiowana jest wersja angielska tekstu. Od tekstu `<plk>` do następnego znacznika (lub do końca tekstu) definiowana jest wersja polska.

Proszę zwrócić uwagę, że w znacznikach korzystamy z identyfikatorów języków tych samych, które zostały wymienione w definicji `languages` sekcji `environ`.

Wersje te mogą być przemieszane, jak pokazuje następujący przykład:

```
name = "<enu>Age<plk>Wiek<enu> 15-24<plk> 15-24";
```

O ile powyższy przykład jest nieco sztuczny, zwraca uwagę na jedno: pojawia się część napisu, którą w takiej samej postaci chcielibyśmy umieścić we wszystkich językach. Można to zrobić jak niżej:

```
name = "<enu>Age<plk>Wiek<*> 15-24";
```

Czyli tekst od znacznika `<*>` do następnego znacznika będzie wyświetlany we wszystkich wersjach językowych.

Jeżeli na początku tekstu nie podamy żadnego znacznika, tylko od razu wpisujemy tekst:

```
name = "Resp: <enu>sex<plk>płeć";
```

program traktuje to tak, jakby tekst ten zaczynał się od znacznika `<*>` (czyli w powyższym przykładzie tekst "Resp: " będzie wyświetlany w obu wersjach językowych).

4.6 Survey (badanie)

W definicji tej opisujemy podstawowe informacje o badaniu:

- `demo` (wersja demonstracyjna) [opcjonalnie]

  ```
  "demo" "=" wartość_logiczna ";"
  ```

  Dla wersji demonstracyjnej badania, aplikacja YAC Data Analyzer będzie wyświetlała ostrzeżenie, że dane nie są reprezentatywne i nie należy na ich podstawie prowadzić wnioskowań statystycznych.

- `excludeSysMis` (wyłączenie systemowych braków danych z obliczeń) [opcjonalnie]

  ```
  "excludeSysMis" "=" wartość_logiczna ";"
  ```

  Pozwala włączyć lub wyłączyć uwzględnianie systemowych braków danych do podstaw obliczeń. Np. mając odpowiedzi na pytanie: tak (3 razy), nie (raz), brak danych (raz), procentowanie przy włączonych brakach danych i wyłączonych będzie wyglądało odpowiednio: 60%, 20% i 75%, 25% (w pierwszym przypadku procenty nie sumują się do 100, gdyż brak danych jest uwzględniany w podstawie obliczeń).

Uwaga 1: jeżeli instrukcja nie zostanie podana, braki danych będą domyślnie włączane do obliczeń.
Uwaga 2: w plikach badań systemowym brakiem danych będzie każdy tekst nie przedstawiający sobą żadnej liczby (więc np. same spacje).

- `expires` [opcjonalnie]

  ```
  "expires" "=" rok [ "/" miesiąc [ "/" dzień ] ] ";"
  ```

  Określa datę, do której badanie będzie dostępne.

  Uwaga
  Domyślną wartością `miesiąca` jest 1 (styczeń); także domyślną wartością `dnia` jest 1. Jeżeli więc w dacie zostanie podany tylko rok, np. 2010, badanie będzie aktywne tylko do pierwszego stycznia, 2010.

- `fixedRecordsets` [opcjonalnie]

  ```
  "fixedRecordsets" "=" wartość_logiczna ";"
  ```

  Blokuje możliwość przełączenia zbioru danych w analizach aplikacji YAC Data Analyzer.

- `hideRowsCols` [opcjonalnie]

  ```
  "hideRowsCols" "=" "off" | "base" | "values" ";"
  ```

  Określa domyślną obsługę pustych wierszy i kolumn w aplikacji YAC Data Analyzer:
  - `off` - puste wierszy i kolumny nie będą chowane,
  - `base` - wiersze i kolumny o zerowych podstawach będą chowane,
  - `values` - wiersza i kolumny z zerowymi wartościami będą chowane.

- `icon` (ikona badania) [opcjonalnie]

  ```
  "icon" "=" nazwa_pliku ";"
  ```

  Jeżeli z badaniem związane jest logo lub inne oznaczenie, można dodać do badania ikonę, która będzie wyświetlana w programie YAC Data Analyzer i Eksploratorze Windows. Nazwę pliku należy podać w cudzysłowie.

- `id` (identyfikator badania)

  ```
  "id" "=" identyfikator ";"
  ```

  Definicja ta służy do jednoznacznego określania poszczególnych badań dystrybuowanych przez jedną firmę badawczą. Żadne dwa badania (w ramach tej samej przestrzeni nazw) nie powinny mieć takiego samego identyfikatora.

  Jest to o tyle ważne, że program YAC Data Analyzer zapisując definicje parametrów czy raportów, zapisuje identyfikator badania, do którego te definicje należą.

- `logo` (logo firmy lub badania) [opcjonalnie]

  ```
  "logo" "=" nazwa_pliku ";"
  ```

  Podany obrazek (musi być to bitmapa Windows lub plik w formacie JPEG lub GIF) program YAC Data Analyzer będzie wyświetlał w pasku stanu lub w menedżerze badania. Może być to logo firmy dystrybuującej badanie, logo badania lub dowolna inny obrazek.

  Pasek stanu ma obecnie 30 pikseli wysokości - obrazki wyższe będą przeskalowane tak, aby się w tej wysokości zmieściły. Szerokość jest dowolna (choć nie powinna być na tyle duża aby nie zostawić miejsca na informacje wyświetlane w pasku stanu).

  Nie ma ograniczeń co do wielkości obrazka wyświetlanego w menedżerze badania.

Lewy dolny piksel (lub rzadziej, prawy górny - zależy od formatu obrazka) przedstawia kolor przezroczysty - ten kolor nie będzie wyświetlany (będzie widać tło - kolor paska stanu lub menedżera badania).

- `logoPlacement` (pozycja logo firmy lub badania) [opcjonalnie]

  ```
  "logoPlacement" "=" "statusBar" | "surveyManager" ";"
  ```

  Podany w instrukcji `logo` obrazek będzie wyświetlany odpowiednio: w pasku stanu lub w oknie menedżera badania.

- `name` (nazwa badania)

  ```
  "name" "=" tekst_jednolinijkowy ";"
  ```

  Definicja nazwy opisowej badania.

- `namespace` (przestrzeń nazw)

  ```
  "namespace" "=" identyfikator ";"
  ```

  Definicja ta służy do określenia firmy dystrybuującej badanie. Identyfikator ten powinien być taki sam dla wszystkich badań jednej firmy badawczej. Prosimy o kontakt z firmą YAC Software w celu określenia tego identyfikatora dla Państwa firmy.

- `owner` (właściciel badania) [opcjonalnie]

  ```
  "owner" "=" tekst_jednolinijkowy ";"
  ```

  Tekst ten będzie wyświetlany w programie YAC Data Analyzer w menedżerze badania.

- `stdStats` (statystyki standardowe) [opcjonalne]
  `advStats` (statystyki zaawansowane) [opcjonalne]

  ```
  "stdStats" "=" identyfikator { "," identyfikator } ";"
  "advStats" "=" identyfikator { "," identyfikator } ";"
  ```

  Pola te definiują grupy statystyk standardowych i zaawansowanych - grupy te będą wyświetlane w oddzielnych zakładkach w oknie dialogowym parametru statystyk w programie YAC Data Analyzer.

  Dopuszczalne identyfikatory pojedynczych statystyk:

- `cntA` - Liczebność faktyczna - liczba (faktyczna) respondentów w komórce.
- `cntW` - Liczebność ważona - liczba (ważona) respondentów w komórce.
- `pctLayer` - Procent w warstwie - liczba respondentów w komórce podzielona przez liczbę respondentów w warstwie.
- `pctRow` - Procent w wierszach - liczba respondentów w komórce podzielona przez liczbę respondentów w wierszu.
- `pctCol` - Procent w kolumnach - liczba respondentów w komórce podzielona przez liczbę respondentów w kolumnie.
- `index` - Indeks podobieństwa - podobieństwo wyników w komórce do podstawy pokazane jako indeks (wartość komórki dzielona przez wartość oczekiwaną na podstawie rozkładów brzegowych pomnożona przez 100).
- `est` - Estymacja populacyjna - estymowana wielkość populacji w komórce w tysiącach osób.
- `meanRow` - Średnia w wierszach - średnie z komórek w wierszu grupowane po pytaniach.
- `meanCol` - Średnia w kolumnach - średnie z komórek w kolumnie grupowane po pytaniach.
- `varRow` - Wariancja w wierszach - wariancje z komórek w wierszu grupowane po pytaniach.
- `varCol` - Wariancja w kolumnach - wariancje z komórek w kolumnie grupowane po pytaniach.
- `stdDevRow` - Odchylenie standardowe w wierszach - odchylenia standardowe z komórek w wierszu grupowane po pytaniach.
- `stdDevCol` - Odchylenie standardowe w kolumnach - odchylenia standardowe z komórek w kolumnie grupowane po pytaniach.
- `sumRow` - Suma w wierszach - sumy komórek w wierszu grupowane po pytaniach.
- `sumCol` - Suma w kolumnach - sumy komórek w kolumnie grupowane po pytaniach.
- `shareRow` - Udział w wierszach - udział w wierszu sumy komórek w kolumnie, grupowany po pytaniach.
- `shareCol` - Udział w kolumnach - udział w kolumnie sumy komórek w wierszu, grupowany po pytaniach.
- `indLayer` - Wskaźnik w warstwie - wartości wskaźników medialnych w warstwie.
- `indRow` - Wskaźnik w wierszach - wartości wskaźników medialnych w wierszach.
- `indCol` - Wskaźnik w kolumnach - wartości wskaźników medialnych w kolumnach.

Można też użyć następujących identyfikatorów do zdefiniowania grup statystyk:
- `cnt` - liczebności.
- `pct` - procenty.
- `mean` - średnie.
- `var` - wariancje.
- `stdDev` - odchylenia standardowe.
- `sum` - sumy.
- `share` - udziały.
- `ind` - wskaźniki medialne.

Uwaga
Jeżeli pola te nie zostaną zdefiniowane, wszystkie statystyki będą traktowane jako statystyki standardowe.

Uwaga
Jeżeli zostaną zdefiniowane tylko statystyki standardowe, tylko te statystyki będą dostępne w programie YAC Data Analyzer; jeżeli zostaną zdefiniowane tylko statystyki zaawansowane, wszystkie pozostałe statystyki będą traktowane jako standardowe.

- `wizards` (kreatory analiz)

  ```
  "wizards" "=" identyfikator { "," identyfikator } ";"
  ```

  W tym polu określamy, jakie kreatory analiz będą dostępna dla badania. Obecnie można zdefiniować następujące kreatory (w pierwszej kolumnie podane są ich identyfikatory):

  Analizy ogólne:
  - `table1d` tabele jednowymiarowe,
  - `table2d` tabele dwuwymiarowe,
  - `cmpWaves` porównanie fal,
  - `cmpGroups2` porównanie dwóch grup docelowych (prostsza wersja kreatora `cmpGroups`),
  - `cmpGroups` porównanie grup docelowych,
  - `genComplex` kreator analiz bazujących na pytaniach złożonych (wielowymiarowych),
  - `genCrossBann` tabele dwuwymiarowe z procentami i średnimi.

  Jeżeli chcemy udostępnić wszystkie powyższe kreatory, wystarczy podać jeden identyfikator: `general`.

  Przykłady:
  ```
  wizards = table1d, table2d, cmpGroups;
  wizards = general;
  ```

Przykład całej definicji z użytymi dotychczas polami:

```
def survey
  namespace = yac.com.pl;
  id = demo;
  owner = "YAC Software";
  name = "<enu>Demonstration survey<plk>Badanie demonstracyjne";
  demo = 1;
  icon = "Demo.ico";
  logo = "CompanyImage.jpg";
  logoPlacement = surveyManager;
  wizards = general;
  excludeSysMis = 1;
  hideRowsCols = values;
end;
```

4.6.1 Infopage (strony informacyjne)

W definicji badania można jeszcze opcjonalnie dodać strony informacyjne o badaniu. Można tu opisać np. metodologię doboru próby, strukturę próby, kwestionariusz, materiały pomocnicze, harmonogram realizacji, historię projektu i tym podobne informacje.

Służy do tego złożona definicja `infopage`, w której mogą znaleźć się następujące pola:

- `name` (nazwa strony)

  ```
  "name" "=" tekst_jednolinijkowy ";"
  ```

  Nazwa strony wyświetlana w spisie treści w Menedżerze Badania programu YAC Data Analyzer.

- `text` (tekst strony)

  ```
  "text" "=" tekst_wielolinijkowy ";"
  ```

  Tekst, który zostanie wyświetlony po wyborze tej strony w spisie treści. W tekście można korzystać ze wszystkich znaczników i opcji języka HTML.

- `file` (plik informacyjny strony)

  `"file" "=" tekst_jednolinijkowy ";"`

  Zamiast wpisywać tekst strony do dokumentacji badania można podać nazwę pliku, w którym opis ten się znajduje. Plik może być w dowolnym formacie "rozumianym" przez Windows (tzn. aplikacja do jego czytania musi być zarejestrowana w systemie). Takimi plikami są np. pliki w formacie HTML, pliki tekstowe, pliki popularnych aplikacji jak Word czy Excel.

  Jeżeli w systemie nie jest zarejestrowany odpowiedni program do czytania pliku, w programie YAC Data Analyzer zostanie wyświetlony komunikat o niemożności wyświetlenia strony.

- `addr` (adres strony)

  `"addr" "=" tekst_jednolinijkowy ";"`

  Pliki dystrybuowane z badaniem może być trudno zaktualizować. Można wtedy informacje o badaniu umieścić na własnych stronach internetowych, a w dokumentacji badania podać adres do tych stron. Wtedy klient łącząc się z tymi stronami może mieć dostęp do najbardziej aktualnych informacji o badaniu.

  Program przy przetwarzaniu tej instrukcji sprawdza, czy taka strona rzeczywiście w internecie istnieje.

- `openInNewWindow` (otwórz w nowym oknie)

  `"openInNewWindow" "=" wartość_logiczna ";"`

  Domyślnie strona informacyjna wyświetlana jest w programie YAC Data Analyzer (w oknie **Menedżer Badania**). Jeżeli chcemy aby strona ta została otwarta i wyświetlona w oknie odpowiadającej jej aplikacji (np. w MS Word), to należy zdefiniować powyższą opcję z wartością 1.

Przykład całej definicji z użytymi dotychczas polami:

```
def survey
  . . .
  def infopage
    name = "<enu>Survey calendar<plk>Harmonogram badania";
    file = "<enu>Calendar.xls<plk>Kalendarz.xls";
    openInNewWindow = 1;
    def infopage
      name = "<enu>Vacations<plk>Wakacje";
      text = <<
        <enu> During summer vacations <b>no</b> interviews will be
              conducted.
        <plk> W czasie letnich wakacji <b>nie</b> będą przeprowadzane
              żadne wywiady.
      >>;
    end;
  end;
  def infopage
    name = "<enu>Current wave status<plk>Stan realizacji obecnej fali";
    addr = "yac.com.pl/<enu>curWaveStatus<plk>stanAktFali<*>.html";
  end;
end;
```

Uwagi
- Strony informacyjne mogą być zagnieżdżane (podtematy).
- Nazwy plików, tekst strony, adres - mogą być wersjonowane językowo.
- Wielokrotne spacje i łamanie tekstu w tekstach wielolinijkowych są ignorowane przez program YAC Data Analyzer. Sposób formatowania długich tekstów w powyższym przykładzie służy tylko zwiększeniu czytelności dokumentacji.
- W tekstach wielolinijkowych można używać znaczników HTML - w powyższym przykładzie słowa "no" oraz "nie" na stronie "Wakacje" zostaną pogrubione.
- Aby pokazać kalendarz badania, zostanie uruchomiony program MS Excel, w którym następnie zostanie otwarty odpowiedni plik. Pozostałe strony zostaną pokazane w oknie menedżera badań.
- Dla jednej strony informacyjnej, dozwolona jest tylko jedna z definicji: `text`, `file`, `addr`.

4.7 Licenses (licencje)

Definicja pozwala na włączenie ochrony badań i konfigurację osób uprawnionych do korzystania z danych.

Uwaga
Aby móc chronić badania przed nielegalnych wykorzystaniem, należy dokupić w firmie YAC Software odpowiednie rozszerzenie aplikacji YAC Data Builder.

Opisy poszczególnych licencji umieszcza się w dodatkowych definicjach złożonych. W definicjach poszczególnych licencji nie występują elementy standardowe.

Uwaga
Cała ta sekcja jest opcjonalna. Jeżeli nie zostanie zdefiniowana, przyjmuje się, że dane nie będą chronione.

Przykład:

```
def licenses
  batchCode = "DEMOMA";
  def license
    key = "29E2-4F60-083E-055E-BC23-36C5-1D74-B933-0857";
    type = standAlone;
    expires = 2006/12/20;
  end;
  def license
    key = "4C74-F80F-4E3D-2A37-7F7F-2EE8-21C5-CD58-0936";
    type = network;
```

```
    count = 3;
  end;
  def license
    key = "23728059";
    type = hardwareKey;
    expires = 2007/01/20;
  end;
end;
```

Uwagi
- Powyższa definicja udostępnia dane do korzystania na trzech instalacjach.
- Pierwsza licencja jest licencją jednostanowiskową, która wygasa z dniem 2006-12-20.
- Druga licencja jest licencją sieciową na 3 jednoczesne uruchomienia, która nigdy nie wygasa.
- Trzecia licencja oparta jest o klucz sprzętowy i wygasa 2007-01-20.
- Pierwsze dwie licencje oparte są na oprogramowaniu YAC License Kit; ostatnia licencja wykorzystuje klucze sprzętowe Aladdin HASP.
- Jeżeli definiujesz licencje oparte na kluczach sprzętowych, Twój Batch Code (dostarczany przez firmę Aladdin) musi zostać zdefiniowany w oddzielnej instrukcji `batchCode`. Ten kod zostanie wykorzystany do wczytania Twojego Vendor Code z pliku `.hvc`. Plik ten musi znajdować się albo w domyślnym katalogu Aladdina, lub w tym samym katalogu co plik badania `.dbs`. Nazwa pliku powinna mieć postać `<batchCode>.hvc` (w powyższym przypadku: `DEMOMA.hvc`).

4.8 Recordset (zbiór danych)

Zbiory danych służą do zdefiniowania danych źródłowych dla badania.

Definicja zawiera pola standardowe (`id`, `name`, `nick`, `info`).

Dozwolone są także następujące definicje:

- `hidden` (ukryty) [opcjonalnie]

  `"hidden" "=" wartość_logiczna ";"`

  Blokuje możliwość wyboru tego zbioru jako podstawy obliczeń w analizach aplikacji YAC Data Analyzer. Jednak badanie może nadal składać się z pytań należących do wielu zbiorów danych w relacji jeden-do-wielu).

- `stats` (statystyki) [opcjonalne]

  `"stats" "=" identyfikator { "," identyfikator } ";"`

  Ogranicza dostępne statystyki dla analiz opartych na tym zbiorze danych.

  Dopuszczalne identyfikatory dla tej instrukcji opisane są w punkcie `stdStats / advStats` dla sekcji `survey`.

Uwaga
Pole nick wykorzystywane jest tylko w przypadku badań, w których definiuje się wiele zbiorów danych; skrócona nazwa wyświetlana jest jako wartość w parametrze **Podstawa obliczeń**.

Ponadto definicja ta zawiera definicje poszczególnych plików danych wchodzących w skład całego zbioru.

4.8.1 Files (pliki)

Jest to lista definicji postaci:

```
"colSpecFormat" "=" identyfikator_formatu ";"
"colSpec" "=" nazwa_pliku_opisowego ";"
"dataFile" "=" nazwa_pliku_danych ";"
```

W pliku opisowym (`colSpec`) znajduje się specyfikacja kolumn: lokalizacji (ich pozycji w rekordzie i szerokości), nazw, typów, itp. W pliku danych (`dataFile`) zawarte są dane badania. Nazwy plików należy podać w cudzysłowach jako tekst.

Instrukcja `colSpecFormat` określa format w jakim opisane są lokalizacje kolumn (w plikach dalej zdefiniowanych za pomocą instrukcji `colSpec`). Po instrukcji `colSpecFormat` należy podać jeden z następujących identyfikatorów:
- `simple` (prosty opis kolumn),
- `sps` (opis kolumn zgodny z plikami SPSS Syntax).

Przykłady plików opisowych w powyższych formatach znajdują się w aneksie B.

Definicja plików może zawierać wiele takich elementów (`colSpecFormat`, `colSpec`, `dataFile`). Definicja formatu kolumn dotyczy wszystkich kolejnych plików opisowych, chyba że zostanie powtórzona z innym identyfikatorem. Definicja lokalizacji kolumn dotyczy wszystkich kolejnych plików danych, aż do następnej definicji lokalizacji kolumn. Definicja plików powinna się zatem zaczynać od definicji formatu i lokalizacji kolumn, a kończyć na definicji pliku danych.

Jeżeli format lokalizacji kolumn nie zostanie podany, przyjmuje się, że jest to format prosty (`simple`).

Przykład:

```
def recordset
  id = respondents;
  name = "<enu>respondents<plk>respondenci";
  nick = "Resp:";
  info = "<enu>Respondents' background data<plk>Metryczka respondenta";
  def files
    colSpecFormat = sps;
    colspec = "fdemo001.var";
    datafile = "fdemo001.dat";
    datafile = "fdemo002.dat";
    colspec = "fdemo003.var";
    datafile = "fdemo003.dat";
    datafile = "fdemo004.dat";
  end;
end;
```

Uwagi
- Definicje kolumn dla pierwszych dwóch plików danych są takie same i są zawarte w pliku `fdemo001.var`. Podobnie kolejne dwa pliki mają takie same definicje kolumn (opisane w pliku `fdemo003.var`).
- Wszystkie pliki definiujące kolumny zapisane są w formacie SPSS Syntax.
- Zawartość pola info (jeżeli zdefiniowane) będzie wyświetlana w pierwszym oknie kreatora analiz.
- Pliki danych powinny być w formacie Fixed-ASCII.

4.8.2 Join (złączenie)

W badaniach opartych na kilku zbiorach danych korzysta się z definicji `join` do wskazania, po których kolumnach zbiory te będą łączone.

```
def recordset
  . . .
  def files
    . . .
  end;
  join = id;
end;
```

Oznacza to, że definiowany zbiór danych zostanie połączony z głównym zbiorem danych (definiowanym jako pierwszy) po kolumnie `id`.

Uwagi
- Kolumna `id` musi istnieć w tym zbiorze i w pierwszym zbiorze; musi także istnieć we wszystkich plikach tych zbiorów danych.
- Instrukcja `join` nie ma sensu dla pierwszego zbioru danych, jest obowiązkowa dla kolejnych zbiorów danych.

4.9 Weights (wagi)

Definicja pozwala na określenie wag analitycznych w badaniu.

Opcja udostępniania obliczeń na danych nie ważonych:
`"noweight" "=" wartość_logiczna ";"`

Opisy poszczególnych wag analitycznych umieszcza się w dodatkowych definicjach złożonych. W definicjach poszczególnych wag występują elementy standardowe. Ponadto należy zdefiniować, z której kolumny w danych będą brane wartości wagi:
`"column" "=" nazwa_kolumny ";"`

Podstawa estymacji na populację może być zdefiniowana zarówno dla danych ważonych jak i nie ważonych. Jeżeli nie zostanie zdefiniowana, estymacje na populację nie będą możliwe. Służy do tego instrukcja `population`.

Uwaga
Cała ta sekcja jest opcjonalna. Jeżeli nie zostanie zdefiniowana, przyjmuje się, że udostępnione będą obliczenia na danych nie ważonych.

Przykład:

```
def weights
  noweight = 1;
  def weight
    id = population;
    name = "<enu>population<plk>populacyjna";
    column = v17;
    population = 29957930;
  end;
  def weight
    id = households;
    name = "<enu>households<plk>gosp. domowych";
    column = v18;
    population = 13582900;
  end;
end;
```

Uwagi
- Powyższa definicja udostępnia dwie wagi oraz obliczenia na danych nie ważonych. Wagi brane są odpowiednio z kolumn `v17` i `v18` plików danych.

- Estymacje na populację będą dostępne tylko dla danych ważonych.
- Zawartość pola `info` (jeżeli zdefiniowane) będzie wyświetlana w pierwszym oknie kreatora analiz.

4.10 Waves (fale)

W definicji `waves` (fale) należy zdefiniować poszczególne fale; dopuszczalna jest także definicja:

- `minSelCount` [opcjonalnie]

  ```
  "minSelCount" "=" liczba_całkowita ";"
  ```

  Określa minimalną liczbę fal potrzebnych do analizy; jeżeli użytkownik programu YAC Data Analyzer wybierze mniejszą liczbę fal, obliczenia nie zostaną przeprowadzone.

W definicjach poszczególnych fal występują elementy standardowe. Ponadto należy zdefiniować daty realizacji fal: datę początkową i (opcjonalnie) datę końcową:

- `dateStart` (data początkowa) [obowiązkowe]

  ```
  "dateStart" "=" rok [ "/" miesiąc [ "/" dzień ] ] ";"
  ```

  Definiuje początek fali.

  Rok, miesiąc oraz dzień są liczbami ze standardowych zakresów. Rok należy podawać jako liczbę czterocyfrową. Jeżeli data końcowa nie zostanie zdefiniowana, przyjmuje się, że jest ona równa dacie początkowej. Podawanie miesiąca i dnia jest opcjonalne (ale aby podać dzień, należy podać miesiąc).

  Format zapisu daty jest niezależny od ustawień regionalnych Windows. W programie YAC Data Analyzer natomiast będą wyświetlane zgodnie z ustawieniami regionalnymi.

- `dateEnd` (data końcowa) [opcjonalne]

  ```
  "dateEnd" "=" rok [ "/" miesiąc [ "/" dzień ] ] ";"
  ```

  Uwaga
  Jeżeli zdefiniowana, musi określać datę późniejszą niż `dateStart`.

Przykład:

```
def waves
  minSelCount = 2;
```

```
  def wave
    name = "Pilotaż";
    dateStart = 2002/01;
  end;
  def wave
    name = "pretest";
    dateStart = 2002/02/01;
    dateEnd = 2002/02/14
  end;
  def wave
    name = "posttest";
    dateStart = 2002/06/01;
    dateEnd = 2002/06/14;
  end;
end;
```

Uwagi

- Jeżeli nie zostanie podany identyfikator fali, przyjmuje się, że jego postać to `waveX`, gdzie `X` to kolejny numer fali.
- Nazwa fali wyświetlana jest np. w dialogu wyboru fal w programie YAC Data Analyzer. Jeżeli nie zostanie podana, przyjmuje się, że jej postać to Wave X lub Fala X, gdzie X to kolejny numer fali.
- Zdefiniowanych fal powinno być tyle samo, co plików w definicji zbioru danych. Jest to rozwiązanie tymczasowe i zostanie usunięte w jednej z najbliższych wersji programu.

4.11 Data (dane)

W tej definicji złożonej opisuje się pytania wraz z odpowiedziami, jak również moduły pytań:

```
def data
  // tu następują definicje pytań i / lub modułów
end;
```

4.11.1 Module (moduł)

Moduły służą do organizacji dostępnych w badaniu informacji. Oprócz standardowych pól, w `module` mogą występować definicje innych modułów oraz definicje pytań.

Moduły oraz występujące w nich pod-moduły i pytania tworzą zatem hierarchię podobną do folderów, pod-folderów i plików znajdujących się na komputerze. Wprowadzanie modułów służy do ułatwienia użytkownikom badania dostępu do poszczególnych pytań. Dlatego też nie powinno się przesadzać ze stopniem zagnieżdżania modułów.

Przykład definicji modułu i pod-modułów:

```
def module
  name = "Metryczka";
  def module
    name = "Respondent";
    info = "Dane metryczkowe respondenta";
    // dalej następują definicje pytań i / lub pod-modułów
  end;
  def module
    name = "Gosp. domowe";
    info = "Dane metryczkowe gospodarstwa domowego";
    // dalej następują definicje pytań i / lub pod-modułów
  end;
end;
```

4.11.2 Question (pytanie)

W modułach można umieścić definicje poszczególnych pytań. Pytania mogą być też umieszczone w definicji danych (`data`) poza definicjami modułów - wówczas pytanie to będzie dostępne w programie YAC Data Analyzer na najwyższym poziomie.

Przykład definicji pytania:

```
def question
  id = sex;
  name = "Płeć";
  info = "Dane wypełnione na podstawie kart doboru.";
  // dalej następują definicje list odpowiedzi
end;
```

Oprócz standardowych pól mogą tu występować listy odpowiedzi (`responseList`) lub definicje pytania wielowymiarowego (`responseGrid`).

Skale pomiaru
Od wersji 3.03 programu można w pytaniu zdefiniować także skalę pomiaru:

```
scale = <identyfikator_skali>;
```

gdzie identyfikatorami skali są: `nominal` (nominalna), `ordinal` (porządkowa) i `interval` (przedziałowa). Na przykład:

```
def question
  id = educ;
  name = "Wykształcenie";
  scale = ordinal;
  // dalej następują definicje list odpowiedzi
end;
```

Jeżeli skala nie zostanie zdefiniowana, przyjmuje się skalę przedziałową.

Dla pytań ze skalą inną niż przedziałowa program analityczny nie pozwala na obliczanie średnich, wariancji i odchyleń standardowych.

4.11.3 ResponseList (lista odpowiedzi)

W każdym pytaniu musi pojawić się definicja odpowiedzi należących do tego pytania. Służy do tego złożona definicja przedstawiona w poniższym przykładzie:

```
def responseList
  // dalej następują definicje poszczególnych odpowiedzi
end;
```

Standardowe pola nie mogą się tu pojawić. W liście odpowiedzi dopuszczalne są definicje odpowiedzi (opisane dalej) oraz proste definicje następujących elementów:

- `column` (kolumna danych)

  ```
  "column" "=" nazwa_kolumny ";"
  ```

  która oznacza, że odpowiedzi będą dotyczyły tej jednej kolumny w danych (najczęściej stosowana w pytaniach typu single-choice). `nazwa_kolumny` jest identyfikatorem i musi się pojawić w co najmniej jednym pliku opisów kolumn wymienionym w definicji zbioru danych (`recordset`).

- `columnList` (lista kolumn danych)

  ```
  "columnList" "=" nazwa_kolumny { "," nazwa_kolumny } ";"
  ```

  która oznacza, że kody wybranych przez respondenta odpowiedzi z pytania multi-choice zapisane zostały na podanych kolumnach (zapis stosowany np. po kodowaniu pytania otwartego).

- `code` (kod odpowiedzi)

  ```
  "code" "=" kod_odpowiedzi ";"
  ```

  która oznacza, że odpowiedzi mają ten sam kod (najczęściej stosowana w pytaniach typu multi-choice). `kod_odpowiedzi` może być dowolną liczbą całkowitą lub rzeczywistą, ewentualnie poprzedzoną znakiem minus.

- `range` (zakres kodów odpowiedzi)

  ```
  "range" "=" lista_kodów_odpowiedzi ";"
  ```

  która oznacza, że w odpowiedziach będą zliczane łącznie podane kody (także stosowana w pytaniach typu multi-choice, gdy na odpowiedź pozytywną składa się więcej niż jeden kod).

- `numeric` (odpowiedzi numeryczne)

  ```
  "numeric" "=" lista_kodów_odpowiedzi ";"
  ```

  która oznacza, że w odpowiedziach będą zliczane kody z podanego zakresu, lecz każdy kod będzie liczony oddzielnie (tzn. przy liczeniu rozkładów odpowiedzi, na jedną taką odpowiedź może przypadać więcej niż jeden wiersz w tabeli - a dokładnie tyle wierszy, ile jest różnych kodów w danych z podanej listy).

  `lista_kodów_odpowiedzi` składa się z poszczególnych kodów i przedziałów kodów, oddzielnych przecinkami. Przedział kodów definiuje się podając wartość minimalną, dwukropek i wartość maksymalną.

  Przykłady:

  ```
  range = 3, 6, 8:9;
  range = 1:2;
  . . .
  numeric = 15:75;
  ```

  Pierwsza instrukcja definiuje odpowiedź, która będzie zliczała łącznie występowanie kodów 3, 6 i od 8 do 9 (np. środek skali, odpowiedź nie dotyczy i braki danych). Druga instrukcja może np. definiować top-two boxes. Natomiast trzecia instrukcja definiuje odpowiedzi numeryczne - każdy kod występujący w danych z zakresu od 15 do 75 zostanie zgłoszony oddzielnie.

  Jako granice przedziałów kodów można podać słowa `min` (brak ograniczenia dolnego) i `max` (brak ograniczenia górnego):

  ```
  numeric = -15:max;   // odpowiedzi od -15 (włącznie) wzwyż
  numeric = min:max;   // bez ograniczeń kodów
  ```

- `attr` (atrybuty odpowiedzi)

  ```
  "attr" "=" lista_identyfikatorów ";"
  ```

  Instrukcja ta opisuje dodatkowe atrybuty odpowiedzi. Obecnie dostępne są:

  - `estIndependent`: estymacje na populacje będą liczone niezależnie dla każdej odpowiedzi; stosowana w przypadku pytań, których odpowiedzi nie dzielą populacji na grupy (jak np. pytanie o dzień tygodnia, w którym wywiad był przeprowadzany - estymacja innych pytań dla każdego dnia tygodnia powinna dawać całą populację),

- `allowOverlaps`: wyłącza kontrolę nachodzenia na siebie kodów odpowiedzi; w poniższym przykładzie program zgłosi ostrzeżenie, że niektóre odpowiedzi mają wspólne kody:

```
def response ... numeric = 0:100; end;
def response ... name = "nie dotyczy"; code = 98; end;
```

  jeżeli do którejś z tych odpowiedzi dodamy atrybut `allowOverlaps`, ostrzeżenie to nie będzie już zgłaszane.

- `noMean`: wyłącza odpowiedź z liczenia średnich; np. załóżmy, że mamy skalę od *zdecydowanie się nie zgadzam* do *zdecydowanie się zgadzam* oraz odpowiedź *nie dotyczy*:

```
def response ... name = "zdecydowanie się nie zgadzam"; code = 1; end;
...
def response ... name = "zdecydowanie się zgadzam"; code = 5; end;
def response ... name = "nie dotyczy"; code = 9; attr = noMean; end;
```

  Dodanie do ostatniej odpowiedzi atrybutu `noMean` spowoduje, że średnia z tego pytania będzie liczona tylko z kodów od 1 do 5 (bez włączania do obliczeń kodu 9).

  Uwaga 1
  Atrybutu nie można używać w odpowiedziach numerycznych (definicja `numeric`).

  Uwaga 2
  W jednym pytaniu, jeżeli dwie odpowiedzi zawierają ten sam kod, nie mogą one definiować `noMean` na różne sposoby, np.:

```
def response ... name = "middle three boxes"; range = 2:4; end;
def response ... name = "ani tak, ani nie"; code = 3; attr = noMean; end;
```

  Obie odpowiedzi zawierają kod 3, ale jedna z nich każe liczyć średnie z tym kodem, a druga to blokuje - definicja nie jest spójna (więc niedozwolona).

  Uwaga 3
  Atrybut ten nie powinien być używany w pytaniach o innej skali pomiaru niż przedziałowa.

- `substStats`: automatycznie przełącza obliczane wyniki w YDA z procentów na średnie.

4.11.4 Response (odpowiedź)

W liście odpowiedzi (responseList) definiujemy poszczególne odpowiedzi za pomocą złożonej definicji, jak pokazuje poniższy przykład:

```
def response
  id = user;
  name = "Użytkownik";
  info = "użytkownik marek wymienionych w pyt. Q1";
  column = v2;
  code = 1;
end;
```

Oprócz standardowych pól mogą tu występować pola attr, column, code, range i numeric, jak w definicji listy odpowiedzi. Pola id oraz name są obowiązkowe. Jedna z definicji: kolumn lub kodów (code, range, numeric) musi wystąpić w definicji odpowiedzi.

Jeżeli pole attr, column lub kodów nie wystąpi w definicji odpowiedzi, używana jest definicja z listy odpowiedzi, do której ta odpowiedź należy. Definicja kolumn i kodów musi się pojawić albo w liście odpowiedzi albo w samej odpowiedzi.

Jeżeli pole attr, column lub kodów wystąpi zarówno w definicji listy odpowiedzi, jak i w definicji samej odpowiedzi, używana jest ta druga definicja (czyli definicja z odpowiedzi przesłania definicję z listy odpowiedzi). Pozwala to w łatwy sposób łączyć w jednym pytaniu odpowiedzi oparte na różnych kolumnach danych i różnych kodach (vide przykład pytania łączącego).

Jeżeli w liście odpowiedzi występuje wiele instrukcji attr, code, range i numeric, dla danej odpowiedzi brana jest ostatnia taka definicja występująca przez odpowiedzią.

4.11.5 ResponseGrid (pytanie wielowymiarowe)

Definicja złożona responseGrid pozwala definiować pytania wielowymiarowe. Poniżej podane są przykłady takich pytań:

Proszę powiedzieć, na ile Pan/i zgadza się z następującymi stwierdzeniami:

	zdecydowanie się zgadzam	*raczej się zgadzam*	*raczej się nie zgadzam*	*zdecydowanie się nie zgadzam*
uwielbiam pizzę				
boję się pająków				
abonament RTV należy znieść				

Tego typu pytaniem będzie także:

Do których marek pasują podane stwierdzenia (możliwość wyboru wielu marek dla każdego stwierdzenia)?

	marka A	*marka B*	*marka C*
dobra goryczka			
ładny kolor			
odpowiednio mocne			
w dobrej cenie			

Więcej przykładów znajduje się w punkcie Pytanie wielowymiarowe.

Ogólnie więc pytania wielowymiarowe składają się z kilku osi (stwierdzenia x skala oraz stwierdzenia x marki z powyższych przykładów). Jednak ogólnie są dwa typy takich pytań: gdy w zmiennych zapisane są odpowiedzi z np. jakiejś skali (jak stopień zgodności z wybranymi stwierdzeniami) lub gdy w zmiennych zapisano wybór danej

kombinacji wartości z osi (jak zgodność tak/nie stwierdzeń z markami).

Przykładem pierwszego typu pytania będzie też stopień zgodności stwierdzeń z markami (3 osie: stwierdzenia x marki x skala ocen). Bateria skal (zgodność z opiniami oceniana np. na skali 11-to punktowej) jest też pytaniem wielowymiarowym.

Proszę też zauważyć, że w pytaniach wielowymiarowych najwyżej jedna oś jest osią odpowiedzi (jak skala), lecz takiej osi może też nie być (jak w drugim przykładzie powyżej).

Definicję pytań wielowymiarowych wprowadzono z dwóch względów:
- łatwiejsza obsługa takich pytań w programie YAC Data Analyzer (i możliwość przedstawienia odpowiedzi w takich właśnie tabelach),
- szybsza i bardziej zwarta definicja tego typu pytań w skryptach programu YAC Data Builder.

Żeby definicja takiego pytania była możliwa, nazwy zmiennych muszą być zgodne z pewnym schematem. Elementom poszczególnych osi (nie będących osią odpowiedzi) musi się dać przypisać teksty tak, aby nazwy zmiennych można było generować na podstawie tych tekstów i danej maski.

Na przykład, załóżmy że mamy pytanie stwierdzenia x marki. Niech będzie ono zapisane na zmiennych dychotomicznych:

```
P10a1 - marka A: stw. 1
P10a2 - marka A: stw. 2
P10b1 - marka B: stw. 1
P10b2 - marka B: stw. 2
```

Wtedy możemy przypisać następujące teksty do stwierdzeń i marek:

```
marka A: "a"
marka B: "b"
stw. 1: "1"
stw. 2: "2"
```

Wtedy nazwy zmiennych zgodne będą z maską: `P10**`, gdzie pierwsza gwiazdka odpowiada przypisanym tekstom pierwszej osi, druga gwiazdka odpowiada przypisanym tekstom drugiej osi. Jeżeli etykiety zmiennych także spełniają powyższy warunek (więc np. są zgodne ze schematem `<marka>: <stwierdzenie>`) zostaną one automatycznie wykorzystane do etykietowania osi pytania.

Jeżeli jedną osią jest oś np. stwierdzeń, drugą osią są odpowiedzi (np. na skali), to powyższy schemat musi być spełniony tylko dla pierwszej osi (wszak zmiennych jest tylko tyle, co stwierdzeń).

Definicja pytania wielowymiarowego składa się instrukcji:

```
def responseGrid
  . . .
end;
```

w której mogą występować następujące pola:

- `columnMask` (maska kolumn)

  ```
  "columnMask" "=" maska_nazw_kolumn ";"
  ```

  służąca do zdefiniowania schematu nazywania zmiennych w pytaniu. W masce musi występować tyle gwiazdek (`*`), ile jest osi, bez uwzględniania osi odpowiedzi.

- `code` (kod odpowiedzi)

  ```
  "code" "=" kod_odpowiedzi ";"
  ```

  która oznacza, że odpowiedzi mają ten sam kod (stosowana w pytaniach typu drugiego - brak skali). `kod_odpowiedzi` może być dowolną liczbą całkowitą lub rzeczywistą, ewentualnie poprzedzoną znakiem minus.

- `range` (zakres kodów odpowiedzi)

  ```
  "range" "=" lista_kodów_odpowiedzi ";"
  ```

  która oznacza, że w odpowiedziach będą zliczane łącznie podane kody (także stosowana w pytaniach typu multi-choice, gdy na odpowiedź pozytywną składa się więcej niż jeden kod). Opis postaci `lista_kodów_odpowiedzi` znajduje się w punkcie ResponesList (lista odpowiedzi).

Poza tymi definicjami należy wprowadzić definicje wszystkich osi (także odpowiedzi).

4.11.6 ResponseAxis (oś)

Osie pozwalają definiować poszczególne wymiary pytania wielowymiarowego:

```
def responseAxis
  . . .
end;
```

W definicji osi mogą wystąpić pola standardowe oraz definicje reponse i responseList, jak w pytaniach. Przy czym:
- tylko odpowiedzi ostatniej osi mogą zawierać instrukcje `range`, `code` lub `numeric`,
- jeżeli w pytaniu została wprowadzona instrukcja `code`, to żadna z odpowiedzi w osiach nie może definiować powyższych instrukcji,
- wszystkie osie, oprócz ostatniej (jeżeli ta definiuje `range`, `code` lub `numeric`), muszą definiować instrukcję `columnText`,
- żadna z odpowiedzi nie może wprowadzać instrukcji `column` czy `columnList`.

Instrukcja `columnText` służy do przypisania odpowiedziom tekstów definiujących nazwy zmiennych. Jej postać to

```
"columnText" "=" tekst_nazw_kolumn ";"
```

więc w przykładzie ze zmiennymi `P10a1..P10b2`, do odpowiedzi "marka A" należy dodać instrukcję

```
columnText = "a";
```

W punkcie Pytanie wielowymiarowe przedstawione zostały podstawowe typy definicji.

4.11.7 Przykłady definicji pytań

W tym punkcie przedstawiamy definicje kilku pytań, które ilustrują wcześniej opisane tematy.

Dla zwiększenia czytelności poniższe pytania (i odpowiedzi na nie) opisane są bez wersji językowych.

4.11.7.1 Pytanie single-choice

Odpowiedzi poniższego pytania oparte są na jednej kolumnie (v3) i jej dwóch wartościach (1 - mężczyźni i 2 - kobiety).

```
def question
  id = sex;
  name = "Płeć";
  def responseList
    column = v3;
    def response id = m; code = 1; name = "mężczyzna"; end;
    def response id = f; code = 2; name = "kobieta"; end;
  end;
end;
```

Pytanie zostało nazwane single-choice, gdyż dla jednej kolumny danych jedna wartość wyklucza pozostałe.

4.11.7.2 Pytanie multi-choice

Odpowiedzi poniższego pytania zostały zapisane jako kolejne kolumny (`nprinter`, `iprinter`, `lprinter`) z wartościami 1 (posiada wyposażenie) i innymi wartościami (nie posiada wyposażenia).

```
def question
  id = peripherals;
  name = "Urządzenia peryferyjne";
  def responseList
    // Poniższy kod identyfikuje odpowiedź pozytywną:
    code = 1;
    def response
      id = pp_nprinter;
      column = nprinter;
      name = "Drukarka igłowa";
    end;
    def response
      id = pp_iprinter;
      column = iprinter;
      name = "Drukarka atramentowa";
    end;
    def response
      id = pp_lprinter;
      column = lprinter;
      name = "Drukarka laserowa";
    end;
  end;
end;
```

Pytanie zostało nazwane multi-choice, gdyż posiadanie jednego z powyższych elementów nie wyklucza posiadania innych.

Pytania multi-choice mogą też być zapisane w inny sposób: na kolejnych zmiennych wymienione są wybrane odpowiedzi (więc nie są to zmienne dychotomiczne; format często stosowany przy kodowaniu pytań otwartych):

```
def question
  id = peripherals;
  name = "Urządzenia peryferyjne";
  def responseList
    // Na tych zmiennych zapisane są kody wybrane przez respondenta:
    columnList = per_1, per_2, per_3;
    def response
      id = pp_nprinter;
      // Kod drukarki igłowej może pojawić się na dowolnej ze wskazanych kolumn.
      // To samo dotyczy kodów pozostałych odpowiedzi.
      code = 1;
      name = "Drukarka igłowa";
    end;
    def response
      id = pp_iprinter;
      code = 2;
      name = "Drukarka atramentowa";
    end;
    def response
      id = pp_lprinter;
      code = 3;
      name = "Drukarka laserowa";
    end;
  end;
end;
```

4.11.7.3 Pytanie łączące

W tym pytaniu są połączone dane z dwóch różnych kolumn. Takie przedstawienie informacji może być w niektórych przypadkach wygodniejsze dla użytkowników badania.

```
def question
  id = user;
  name = "User";
  def responseList
    // Większość odpowiedzi będzie brana z tej kolumny:
    column = v1;
    def response
      id = non.user;
      code = 0;
      name = "non-user";
    end;
    def response
      id = user;
      // Uwaga - ta odpowiedź brana jest z innej kolumny:
      column = v2;
      code = 1;
      name = "user";
    end;
    def response id = light;  code = 1; name = "light";  end;
    def response id = medium; code = 2; name = "medium"; end;
    def response id = heavy;  code = 3; name = "heavy";  end;
  end;
end;
```

W powyższym przykładzie w kolumnie `v1` danych mamy zapisane to, jakim użytkownikiem jest respondent (0 - non-user, 1 - light, 2 - medium, 3 - heavy). Natomiast w kolumnie v2 mamy zapisane to, czy w ogóle jest użytkownikiem (0 - non-user, 1 -user). Łączenie tych dwóch kolumn w odpowiedziach jednego pytania może ułatwić badaczowi analizę danych (ma dostęp jednocześnie do wyników łącznych oraz w rozbiciu na poszczególne kategorie).

Powyższe pytanie można też zdefiniować za pomocą słowa `range`:

```
def question
  id = user;
  name = "User";
  def responseList
    column = v1;
    def response id = non.user; code = 0;     name = "non-user"; end;
    def response id = user;     range = 1:3; name = "user";     end;
    def response id = light;    code = 1;     name = "light";    end;
    def response id = medium;   code = 2;     name = "medium";   end;
    def response id = heavy;    code = 3;     name = "heavy";    end;
  end;
end;
```

I jeszcze jeden przyk³ad z Generalnego Sonda¿u Spo³ecznego (General Social Survey), gdzie wykorzystano ³31czenie kodów do zdefiniowania kategorii wiekowych (pierwsze pytanie pozostawiono np. do dok³adnego okreœlania grup docelowych):

```
def question
  id = agewed;
  name = "Age when first married";
  def responseList
    column = AGEWED;
    def response id = r1; code = 0;       name = "NAP"; attr = noMean; end;
    def response id = rg; numeric = 1:97; name = "age in years"; end;
    def response id = r2; code = 98;      name = "DK"; attr = noMean; end;
    def response id = r3; code = 99;      name = "NA"; attr = noMean; end;
  end;
end;
```

```
def question
  id = agewedg;
  name = "Age when first married (grouped)";
  def responseList
    column = AGEWED;
    def response id = r1;     code = 0;      name = "NAP"; attr = noMean; end;
    def response id = r16m;   range =  1:17; name = "-17"; end;
    def response id = r18_19; range = 18:19; name = "18-19"; end;
    def response id = r20_21; range = 20:21; name = "20-21"; end;
    def response id = r22_25; range = 22:25; name = "22-25"; end;
    def response id = r26_30; range = 26:30; name = "26-30"; end;
    def response id = r31p;   range = 31:97; name = "31-"; end;
    def response id = r2;     code = 98;     name = "DK"; attr = noMean; end;
    def response id = r3;     code = 99;     name = "NA"; attr = noMean; end;
  end;
end;
```

4.11.7.4 Pytanie numeryczne

W tym pytaniu zastosowano słowo `numeric` do zdefiniowania odpowiedzi numerycznych; dodano też odpowiedź "odmowa odpowiedzi".

```
def question
  id = age;
  name = "Age in years";
  def responseList
    column = v2;
    def response id = age;     numeric = 15:75; name = "age";     end;
    def response id = refused; code = 97;       name = "refused"; end;
  end;
end;
```

4.11.7.5 Pytanie wielowymiarowe

Przykład pytania wielowymiarowego stwierdzenia x marki (które stwierdzenia pasują do których marek, możliwość wyboru wielu marek):

```
def question
  id = grid1;
  name = "Stwierdzenia x Marki";
  def responseGrid
    columnMask = "**";
    code = 1;
  def responseAxis
      id = ax1;
      name = "Stwierdzenia";
      def response id = r1; columnText = "p1";
        name = "Produkty tej marki są wyjątkowo smaczne"; end;
      def response id = r2; columnText = "p2";
        name = "Produkty tej marki są wysokiej jakości"; end;
    end;
    def responseAxis
      id = ax2;
      name = "Marki";
      def response id = markaA; columnText = "a"; name = "Marka A"; end;
      def response id = markaB; columnText = "b"; name = "Marka B"; end;
    end;
  end;
end;
```

Zmienne, do których to pytanie się odwołuje to: `p1a`, `p1b`, `p2a`, `p2b`. Zliczany będzie kod 1 (instrukcja `code = 1;`). Instrukcję `code = 1;` można zastąpić np. instrukcją `range = 1:5;`. Wtedy na odpowiedź pozytywną będą składały się kody od 1 do 5.

Pytanie wielowymiarowe stwierdzenia x skala (na ile respondent zgadza się ze stwierdzeniami):

```
def question
  id = grid2;
  name = "Stwierdzenia x Skala";
  def responseGrid
    columnMask = "*";
    def responseAxis
      id = ax1;
      name = "Stwierdzenia";
      def response id = r1; columnText = "p1";
        name = "UPR to najlepsze partia na polskiej scenie politycznej"; end;
      def response id = r2; columnText = "p2";
        name = "Urzędników jest w Polsce za dużo"; end;
    end;
    def responseAxis
      id = ax2;
      name = "Skala";
      def response id = r1; code = 1; name = "zdecydowanie się zgadzam"; end;
      def response id = r2; code = 2; name = "raczej się zgadzam"; end;
      def response id = r3; code = 3; name = "ani się zgadzam, ani nie zgadzam"; end;
      def response id = r4; code = 4; name = "raczej się nie zgadzam"; end;
      def response id = r5; code = 5; name = "zdecydowanie się nie zgadzam"; end;
    end;
  end;
end;
```

`columnText` jest definiowane tylko w pierwszej osi, gdyż druga oś opisuje wartości występujące w kolumnach. Pytanie odwołuje się do zmiennych: `p1`, `p2`. `columnMask` definiuje maskę tylko z jedną gwiazdką. Instrukcja `code` przed definicją osi nie jest w takim przypadku dopuszczalna.

Bateria skal (ocena na skali 11-sto punktowej):

```
def question
  id = grid3;
  name = "Opinie x Ocena";
  def responseGrid
    columnMask = "*";
    def responseAxis
      id = ax1;
      name = "Opinie";
      def response id = r1; columnText = "p1"; name = "Produkt jest za gęsty"; end;
      def response id = r2; columnText = "p2"; name = "Produkt jest za słodki"; end;
    end;
    def responseAxis
      id = ax2;
      name = "Ocena";
      def response id = r1; numeric = 1:11; name = "skala"; end;
      def response id = r2; range = 98,99; name = "trudno powiedzieć / nie dotyczy"; end;
    end;
  end;
end;
```

Dopasowanie marek do stwierdzeń (dla każdej pary: marka - stwierdzenie respondent ma określić, na ile stwierdzenie to pasuje do marki):

```
def question
  id = grid4;
  name = "Stwierdzenia x Marki x Skala";
  def responseGrid
```

```
    columnMask = "**";
    def responseAxis
      id = ax1;
      name = "Stwierdzenia";
      def response id = r1; columnText = "p1";
        name = "Produkty tej marki są wyjątkowo smaczne"; end;
      def response id = r2; columnText = "p2";
        name = "Produkty tej marki są wysokiej jakości"; end;
    end;
    def responseAxis
      id = ax2;
      name = "Marki";
      def response id = markaA; columnText = "a"; name = "Marka A"; end;
      def response id = markaB; columnText = "b"; name = "Marka B"; end;
    end;
    def responseAxis
      id = ax3;
      name = "Skala";
      def response id = r1; code = 1; name = "bardzo pasuje"; end;
      def response id = r2; code = 2; name = "trochę pasuje"; end;
      def response id = r3; code = 3; name = "trochę nie pasuje"; end;
      def response id = r4; code = 4; name = "bardzo nie pasuje"; end;
    end;
  end;
end;
```

4.12 Prasa

Wersja 1.20 aplikacji YAC Data Analyzer wprowadziła możliwość analizy danych czytelnictwa prasy.

W dokumentacji badań dotyczących czytelnictwa prasy może pojawić się nowa sekcja opisująca badane tytuły prasowe, periodyczność, uzyskiwane wskaźniki, itp.

Opis danych prasowych powinien znaleźć się w definicji złożonej

```
def press
  . . .
end;
```

Poza tym dostępne są następujące, dodatkowe kreatory:

- `pressReadership` czytelnictwo pism wg wybranych wskaźników (sortowane alfabetyczne),
- `pressRanking` ranking pism po wybranych wskaźnikach czytelnictwa,
- `pressCmpWaves` trendy czytelnictwa,
- `pressCmpGroups2` porównanie czytelnictwa w dwóch grupach docelowych (prostsza wersja kreatora `pressCmpGroups`),
- `pressCmpGroups` porównanie czytelnictwa w różnych grupach docelowych,
- `pressCoreadership` współczytanie pism,
- `pressStructure` struktura czytelników,
- `pressStructureRow` czytelnictwo przecięte przez wybrane pytania,
- `pressPrices` analiza cenników,
- `pressMediaPlan` media-plan,
- `pressMediaPlanOpt` optymalizator media-planów.

Jeżeli chcemy udostępnić wszystkie powyższe kreatory, wystarczy podać jeden identyfikator: `press`.

Uwaga
W wersji Lite nie są dostępne specjalistyczne analizy czytelnictwa prasy.

4.12.1 Definicje wstępne

Na początku definicji danych prasowych należy zdefiniować jedno pole:

- `hook`

  `"hook" "=" nazwa_modułu ";"`

  Wyznacza moduł w hierarchii pytań, w którym znajdą się automatycznie generowane pytania dotyczące prasy.

4.12.2 Indicators (wskaźniki)

W tej sekcji opisywane są badane wskaźniki.

W definicji **`def`** `indicators . . .` **`end;`** należy umieścić definicje poszczególnych wskaźników wg następującego przykładu:

```
def indicator
  id = spo;
  name = "<enu>Spontaneous awareness<plk>Znajomość spontaniczna<*> (%)";
  info = <<
    <enu>Spontaneous awareness of the press title (of the group of press titles)
    <plk>Znajomość spontaniczna tytułów prasowych (zestawu tytułów prasowych)
  >>;
end;
```

W powyższym przykładzie pojawiły się pola standardowe, przy czym identyfikatory wskaźników mają z góry określone znaczenie (powyższe definicje pozwalają zatem określić podzbiór wszystkich wskaźników, które będą dostępne w analizach oraz pozwalają opisać te wskaźniki w językach, w których będzie dostępne badanie).

Identyfikatory wskaźników mogą być następujące:
- `tom` - pierwsze wskazanie (Top of Mind),
- `spo` - znajomość spontaniczna tytułów,
- `awa` - znajomość wspomagana (całkowita) tytułów,
- `distrib` - dotarcie do sklepu,
- `er` - czytane kiedykolwiek,
- `scr` - czytelnictwo cyklu sezonowego,
- `rr` - czytane regularnie,
- `lir` - czytelnictwo ostatniego wydania,
- `nir` - liczba czytanych wydań cyklu sezonowego,
- `mnir` - średnia liczba czytanych wydań cyklu sezonowego,
- `ar` - czytelnictwo przeciętnego wydania,
- `cppg`, `cppr`, `cpper` - Cost Per Point (GRP, Reach, Effective Reach),
- `cpmg`, `cpmr`, `cpmer` - Cost Per Thousand (GRP, Reach, Effective Reach),
- `cpi` - Cost Per Insertion,
- `coi` - Cost Of Insertions,
- `noi` - Number of Insertions,
- `fd` - Opportunities To See,
- `fdplus` - Opportunities To See (plus),
- `af` - Average Frequency,
- `grp` - Gross Rating Points,
- `gi` - Gross Impressions ('000),
- `nci` - liczba kontaktów z wydaniem,
- `mnci` - średnia liczba kontaktów z wydaniem,
- `rpc` - liczba czytelników na kopię,
- `mrpc` - średnia liczba czytelników na kopię,
- `pir` - część czytanego wydania.

Inne identyfikatory nie są dozwolone.

Jako że wskaźniki potrzebują informacji, na których kolumnach znajdują się ich dane, poza polami standardowymi, należy zdefiniować następujące pola:

- `column` (kolumna danych)

  ```
  "column" "=" nazwa_kolumny ";"
  ```

  Wyznacza kolumnę w zbiorze danych, z której będą pobierane dane dla tego wskaźnika (stosujemy, jeżeli wskaźnik korzysta tylko z jednej kolumny w zbiorze (zmienna jednowyborowa)).

- `columnList` (lista kolumn)

  ```
  "columnList" "=" nazwa_kolumny [ "," nazwa_kolumny ] ";"
  ```

  Wyznacza kolumny w zbiorze danych, z których będą pobierane dane dla tego wskaźnika (stosujemy, jeżeli wskaźnik korzysta z wielu kolumn w zbiorze (zmienna wielowyborowa)).

- `numeric` (odpowiedzi numeryczne)

  ```
  "numeric" "=" lista_kodów_odpowiedzi ";"
  ```

  Wyznacza zakres możliwych wartości, oczekiwanych na kolumnach w zbiorze danych, z których będą pobierane dane dla tego wskaźnika (stosujemy, jeżeli wskaźnik korzysta z jednej kolumny lub wielu kolumn w zbiorze).

- `columnMask` (maska kolumn)

  `"columnMask" "=" maska_nazw_kolumn ";"`

  Wyznacza kolumny w zbiorze danych, z których będą pobierane dane dla tego wskaźnika. Instrukcja jest stosowana wtedy, gdy dla każdego pisma dane dla wskaźnika trzymane są w oddzielnej kolumnie.

  W specyfikacji `maska_nazw_kolumn` musi się pojawić znak "*" (gwiazdka). Tworząc nazwę kolumna dla pisma o identyfikatorze A, w miejsce gwiazdki zostanie wstawiony identyfikator A. Na przykład taki zapis może wyglądać następująco:

  ```
  columnMask = "q1_*" ;
  ```

  W tym przykładzie będą dostępne wskaźniki dla kolumn rozpoczynających się "q1_" a dalej będą podstawiane kolejno identyfikatory tytułów prasowych.

- `independentColumns` (kolumny niezależne)

  `"independentColumns" "=" ( 0 | 1 ) ";"`

  Niezależne kolumny definiują jedną kolumnę dla danego pisma w pytaniu multi-choice (zmienne dychotomiczne). Kolumny zależne definiują serię kolumn, w których zapisane są kody poszczególnych pism.

4.12.3 Regions (regiony)

W tej sekcji opisywane są regiony, wg których opisane są zasięgi dystrybucji poszczególnych pism.

W definicji **`def`** `regions . . .` **`end;`** należy umieścić definicje poszczególnych regionów wg następującego przykładu:

```
def region
  id = r01;
  name = "warszawskie";
end;
```

Identyfikatory będą wykorzystywane w dalszej części dokumentacji do opisywania zasięgów pism. Nazwy regionów będą wyświetlane w programie analitycznym.

4.12.4 Titles (pisma)

Pisma opisuje się za pomocą definicji złożonych `module` oraz `title`.

Moduły odgrywają taką samą rolę jak w przypadku definiowania hierarchii pytań - mogą występować wiele razy, w modułach mogą występować definicje kolejnych modułów i / lub pism. Definicje pism mogą też występować poza modułami (więc bezpośrednio w sekcji `press`).

Definicje samych modułów są identyczne do definicji modułów pytań, przejdziemy zatem do opisu pism:

```
def title
  id = TA;
  name = "Pismo A";
  regions = all;
end;
def title
  id = TB;
  name = "Pismo B";
  regions = r01,r19,r35,r63,r69,r93;
end;
```

W powyższym przykładzie zdefiniowano dwa pisma. Oprócz pól standardowych można opisać regiony, w których pismo jest dostępne.

Definicja `regions = all;` oznacza, że pismo dostępne jest we wszystkich regionach (brak definicji regionów równoznaczny jest z dostępnością we wszystkich regionach).

Definicja drugiego pisma określa listę regionów, w których jest dostępne (w przykładzie numeracja jest zgodna z numeracją 49 województw wg GUS). Identyfikatory wykorzystane w definicjach pism muszą być zgodne z identyfikatorami regionów w definicji `regions`.

4.13 Radio

Wersja 1.10 aplikacji YAC Data Analyzer wprowadziła możliwość analizy danych słuchalności radia z badań realizowanych metodą day after recall.

W dokumentacji badań dotyczących słuchalności radia może pojawić się nowa sekcja opisująca badane stacje radiowe, uzyskiwane wskaźniki, itp.

Opis danych radiowych powinien znaleźć się w definicji złożonej

```
def radio
  . . .
end;
```

Poza tym dostępne są następujące, dodatkowe kreatory:

- `radioAudience` słuchalność stacji wg wybranych wskaźników (sortowana alfabetyczne),
- `radioRanking` ranking stacji po wybranych wskaźnikach słuchalności,
- `radioCmpWaves` trendy słuchalności,
- `radioCmpGroups2` porównanie słuchalności w dwóch grupach docelowych (prostsza wersja kreatora `radioCmpGroups`),
- `radioCmpGroups` porównanie słuchalności w różnych grupach docelowych,
- `radioColistening` współsłuchanie stacji,
- `radioStructure` struktura słuchaczy,
- `pressStructureRow` słuchalność przecięta przez wybrane pytania,
- `radioDays` słuchalność stacji po dniach tygodnia,
- `radioQs` słuchalność stacji w kwadransach,
- `radioPlaces` miejsca słuchania stacji,
- `radioSources` źródła sygnału słuchanych stacji.

Jeżeli chcemy udostępnić wszystkie powyższe kreatory, wystarczy podać jeden identyfikator: `radio`.

Uwaga
W wersji Lite nie są dostępne specjalistyczne analizy słuchalności radia.

4.13.1 Definicje wstępne

Na początku definicji danych radiowych należy zdefiniować dwa pola:

- `hook`

  `"hook" "=" nazwa_modułu ";"`

  Wyznacza moduł w hierarchii pytań, w którym znajdą się automatycznie generowane pytania dotyczące radia.

- `dayColumn`

  `"dayColumn" "=" identyfikator_kolumny ";"`

  Wyznacza kolumnę w zbiorze danych, w której zapisany jest dzień tygodnia, o który pytano respondenta.

4.13.2 Indicators (wskaźniki)

W tej sekcji opisywane są badane wskaźniki.

W definicji **`def`** `indicators . . .` **`end;`** należy umieścić definicje poszczególnych wskaźników wg następującego przykładu:

```
def indicator
  id = spo;
  name = "<enu>Spontaneous awareness<plk>Znajomość spontaniczna<*> (%)";
  info = <<
    <enu>Spontaneous awareness of the station (of the group of stations) R
         is equal to the number of respondents who,
         without hearing the list of radio stations,
         declared awareness of the station
         (of at least one station in the group) R.
    <plk>Znajomość spontaniczna stacji (zestawu stacji) R
         jest równa liczbie respondentów,
         którzy bez odsłuchania wykazu stacji radiowych
         zadeklarowali znajomość stacji (co najmniej jednej stacji z zestawu) R.
  >>;
end;
```

W powyższym przykładzie pojawiły się pola standardowe, przy czym identyfikatory wskaźników mają z góry określone znaczenie (powyższe definicje pozwalają zatem określić podzbiór wszystkich wskaźników, które będą dostępne w analizach oraz pozwalają opisać te wskaźniki w językach, w których będzie dostępne badanie).

Identyfikatory wskaźników mogą być następujące:
- `spo` - znajomość spontaniczna stacji,
- `awa` - znajomość wspomagana (całkowita) stacji,
- `week` - zasięg tygodniowy,
- `day` - zasięg dzienny,
- `qs` - zasięg w kwadransach,
- `share` - udział w rynku,
- `meanTime` - średni czas słuchania,
- `avgQ` - audytorium średniego kwadransa.

Inne identyfikatory nie są dozwolone.

4.13.3 Sources (źródła sygnału)

W tej sekcji opisywane są źródła sygnału.

W definicji **`def`** `sources . . .` **`end;`** należy umieścić definicje poszczególnych źródeł wg następującego przykładu:

```
def source
  id = rso;
  name = "<enu>Aerial antenna<plk>Antena naziemna";
end;
```

W powyższym przykładzie pojawiły się pola standardowe, przy czym identyfikatory źródeł sygnału mają z góry określone znaczenie:
- `rso` - antena naziemna,
- `web` - Internet,
- `sat` - antena satelitarna,
- `cab` - TV kablowa.

Inne identyfikatory nie są dozwolone.

4.13.4 Places (miejsca słuchania)

W tej sekcji opisywane są miejsca słuchania.

W definicji **`def`** `places . . .` **`end;`** należy umieścić definicje poszczególnych miejsc wg następującego przykładu:

```
def place
  id = home;
  name = "<enu>At home<plk>W domu";
end;
```

W powyższym przykładzie pojawiły się pola standardowe, przy czym identyfikatory miejsc słuchania mają z góry określone znaczenie:

- `home` - w domu,
- `work` - w pracy,
- `car` - w samochodzie,
- `other` - w innym miejscu.

Inne identyfikatory nie są dozwolone.

4.13.5 Regions (regiony)

W tej sekcji opisywane są regiony, wg których opisane są zasięgi nadajników poszczególnych stacji.

W definicji **`def`** `regions . . .` **`end;`** należy umieścić definicje poszczególnych regionów wg następującego przykładu:

```
def region
  id = r01;
  name = "warszawskie";
end;
```

Identyfikatory będą wykorzystywane w dalszej części dokumentacji do opisywania zasięgów stacji. Nazwy regionów będą wyświetlane w programie analitycznym.

4.13.6 Stations (stacje radiowe)

Stacje radiowe opisuje się za pomocą definicji złożonych `module` oraz `station`.

Moduły odgrywają taką samą rolę jak w przypadku definiowania hierarchii pytań - mogą występować wiele razy, w modułach mogą występować definicje kolejnych modułów i / lub stacji. Definicje stacji mogą też występować poza modułami (więc bezpośrednio w sekcji `radio`).

Definicje samych modułów są identyczne do definicji modułów pytań, przejdziemy zatem do opisu stacji radiowych:

```
def station
  id = S0A;
  name = "Polskie Radio Program 1";
  regions = all;
end;
def station
  id = S1E;
  name = "RADIOSTACJA";
  regions = r01,r19,r35,r63,r69,r93;
end;
```

W powyższym przykładzie zdefiniowano dwie stacje. Oprócz pól standardowych można opisać regiony, w których stacja nadaje.

Definicja `regions = all;` oznacza, że stacja nadaje we wszystkich regionach (brak definicji regionów równoznaczny jest z nadawaniem we wszystkich regionach).

Definicja drugiej stacji określa listę regionów, w których nadaje (w przykładzie numeracja jest zgodna z numeracją 49 województw wg GUS). Identyfikatory wykorzystane w definicjach stacji muszą być zgodne z identyfikatorami regionów w definicji `regions`.

4.14 TV

Wersja 4.00 aplikacji YAC Data Analyzer wprowadziła możliwość analizy danych oglądalności telewizji z badań panelowych.

W dokumentacji badań dotyczących oglądalności telewizji może pojawić się nowa sekcja opisująca badane stacje telewizyjne, uzyskiwane wskaźniki, itp.

Opis danych telewizyjnych powinien znaleźć się w definicji złożonej

```
def tv
  . . .
end;
```

Poza tym dostępne są następujące, dodatkowe kreatory:

- `tvAudience` oglądalność stacji wg wybranych wskaźników (sortowana alfabetyczne),
- `tvRanking` ranking stacji po wybranych wskaźnikach oglądalności,
- `tvCmpWaves` trendy oglądalności,
- `tvCmpGroups2` porównanie oglądalności w dwóch grupach docelowych (prostsza wersja kreatora `tvCmpGroups`),
- `tvCmpGroups` porównanie oglądalności w różnych grupach docelowych,
- `tvCoviewing` współoglądalność stacji,
- `tvStructure` struktura widzów,
- `tvStructureRow` oglądalność przecięta przez wybrane pytania,
- `tvDays` oglądalność stacji po dniach tygodnia,
- `tvQs` oglądalność stacji w kwadransach,
- `tvDaysQs` oglądalność stacji po dniach tygodnia i kwadransach.

Jeżeli chcemy udostępnić wszystkie powyższe kreatory, wystarczy podać jeden identyfikator: `tv`.

Uwaga
W wersji Lite nie są dostępne specjalistyczne analizy oglądalności telewizji.

4.14.1 Definicje wstępne

Na początku definicji danych telewizyjnych należy zdefiniować następujące pola:

- `daysCol`

  ```
  "daysCol" "=" identyfikator_kolumny ";"
  ```

  Wyznacza kolumnę w zbiorze danych, w której zapisany jest dzień tygodnia, o który pytano respondenta (o wartościach od 1 - poniedziałek do 7 - niedziela).

- `minutesCol`

  ```
  "minutesCol" "=" identyfikator_kolumny ";"
  ```

  Wyznacza kolumnę w zbiorze danych, w której zapisana jest liczba minut oglądania danej stacji telewizyjnej.

- `qsCol`

  ```
  "qsCol" "=" identyfikator_kolumny ";"
  ```

  Wyznacza kolumnę w zbiorze danych, w której zapisany jest identyfikator kwadransa (o wartościach 1 - pierwszy kwadrans dnia, 96 - ostatni kwadrans dnia).

- stationsCol

  "stationsCol" "=" identyfikator_kolumny ";"

  Wyznacza kolumnę w zbiorze danych, w której zapisany jest kod stacji telewizyjnej.

4.14.2 Indicators (wskaźniki)

W tej sekcji opisywane są badane wskaźniki.

W definicji **def** indicators . . . **end;** należy umieścić definicje poszczególnych wskaźników wg następującego przykładu:

```
def indicator
  id = amr;
  name = "Average Minute Rating";
  nick = "AMR";
  info = <<
    Average Minute Rating, Średnia Oglądalność Minutowa\n
    \n
    Podstawowy wskaźnik określający wielkość widowni
    oglądającej dany program lub kanał telewizyjny w określonym odcinku czasu.
  >>;
end;
```

W powyższym przykładzie pojawiły się pola standardowe, przy czym identyfikatory wskaźników mają z góry określone znaczenie (powyższe definicje pozwalają zatem określić podzbiór wszystkich wskaźników, które będą dostępne w analizach oraz pozwalają opisać te wskaźniki w językach, w których będzie dostępne badanie).

Identyfikatory wskaźników mogą być następujące:
- amr - Average Minute Rating (Średnia Oglądalność Minutowa),
- atv - Average Time Viewing (Średni Czas Oglądania),
- shr - share (udział),
- rch - reach (zasięg).

Inne identyfikatory nie są dozwolone.

4.14.3 Stations (stacje telewizyjne)

Stacje telewizyjne opisuje się za pomocą definicji złożonych module oraz station.

Moduły odgrywają taką samą rolę jak w przypadku definiowania hierarchii pytań - mogą występować wiele razy, w modułach mogą występować definicje kolejnych modułów i / lub stacji. Definicje stacji mogą też występować poza modułami (więc bezpośrednio w sekcji tv).

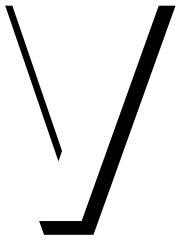
Definicje samych modułów są identyczne do definicji modułów pytań, przejdziemy zatem do opisu stacji telewizyjnych:

```
def station
  id = tvp1;
  name = "TVP1";
  code = 1;
end;
def station
  id = tv4;
  name = "TV 4";
  code = 6;
end;
```

W powyższym przykładzie zdefiniowano dwie stacje.

Oprócz pól standardowych należy zdefiniować kod stacji, który zapisany jest w kolumnie stationsCol danych.

Rozdział



5 Aneksy

5.1 Aneks A - identyfikatory języków

afk	Afrykanerski
sqi	Albański
ena	Angielski (Australia)
enl	Angielski (Belize)
enp	Angielski (Filipiny)
eni	Angielski (Irlandia)
enj	Angielski (Jamajka)
enc	Angielski (Kanada)
enb	Angielski (Karaiby)
enz	Angielski (Nowa Zelandia)
ens	Angielski (Rep. Południowej Afryki)
enu	Angielski (Stany Zjednoczone)
ent	Angielski (Trynidad)
eng	Angielski (Wielka Brytania)
enw	Angielski (Zimbabwe)
arg	Arabski (Algieria)
ara	Arabski (Arabia Saudyjska)
arh	Arabski (Bahrajn)
are	Arabski (Egipt)
ari	Arabski (Irak)
ary	Arabski (Jemen)
arj	Arabski (Jordania)
arq	Arabski (Katar)
ark	Arabski (Kuwejt)
arb	Arabski (Liban)
arl	Arabski (Libia)
arm	Arabski (Maroko)
aro	Arabski (Oman)
ars	Arabski (Syria)
art	Arabski (Tunezja)
aru	Arabski (Zjedn. Emiraty Arabskie)
hye	Armeński
aze	Azerski (cyrylica)
aze	Azerski (łaciński)
euq	Baskijski
bng	Bengalski (Indie)
bel	Białoruski
bsb	Bośniacki (łaciński, Bośnia i Hercegowina)
bgr	Bułgarski
chs	Chiński (ChRL)
zhh	Chiński (Hongkong SAR)
zhm	Chiński (Makau SAR)
zhi	Chiński (Singapur)
cht	Chiński (Tajwan)
hrb	Chorwacki (Bośnia i Hercegowina)
hrv	Chorwacki
csy	Czeski
div	Divehi
dan	Duński
eti	Estoński
fos	Farerski
far	Farsi
fin	Fiński
frb	Francuski (Belgia)
fra	Francuski (Francja)
frc	Francuski (Kanada)

frl	Francuski (Luksemburg)
frm	Francuski (Monako)
frs	Francuski (Szwajcaria)
glc	Galicyjski
ell	Grecki
kat	Gruziński
guj	Gudżarati
heb	Hebrajski
hin	Hindi
ess	Hiszpański (Argentyna)
esb	Hiszpański (Boliwia)
esl	Hiszpański (Chile)
esf	Hiszpański (Ekwador)
esg	Hiszpański (Gwatemala)
esh	Hiszpański (Honduras)
eso	Hiszpański (Kolumbia)
esc	Hiszpański (Kostaryka)
esm	Hiszpański (Meksyk)
esn	Hiszpański (międzynarodowy)
esi	Hiszpański (Nikaragua)
esa	Hiszpański (Panama)
esz	Hiszpański (Paragwaj)
esr	Hiszpański (Peru)
esu	Hiszpański (Portoryko)
esd	Hiszpański (Republika Dominikańska)
ese	Hiszpański (Salwador)
esp	Hiszpański (tradycyjny)
esy	Hiszpański (Urugwaj)
esv	Hiszpański (Wenezuela)
nlb	Holenderski (Belgia)
nld	Holenderski (Holandia)
ind	Indonezyjski
isl	Islandzki
jpn	Japoński
kan	Kannada
cat	Kataloński
kkz	Kazachski
qub	Keczua (Boliwia)
que	Keczua (Ekwador)
qup	Keczua (Peru)
xho	Khosa
kyr	Kirgiski (cyrylica)
knk	Konkani
kor	Koreański
lth	Litewski
lvi	Łotewski
mki	Macedoński (FYRO)
mym	Malajalam (Indie)
msb	Malajski (Brunei Darussalam)
msl	Malajski (Malezja)
mlt	Maltański
mri	Maori
mar	Marathi
mon	Mongolski (cyrylica)
dea	Niemiecki (Austria)
dec	Niemiecki (Liechtenstein)
del	Niemiecki (Luksemburg)
deu	Niemiecki (Niemcy)
des	Niemiecki (Szwajcaria)
nor	Norweski (Bokmal)
non	Norweski (Nynorsk)
pan	Pendżabski

plk	Polski
ptb	Portugalski (Brazylia)
ptg	Portugalski (Portugalia)
rus	Rosyjski
rom	Rumuński
sma	Sami Płd. (Norwegia)
smb	Sami Płd. (Szwecja)
smg	Sami Płn. (Finlandia)
sme	Sami Płn. (Norwegia)
smf	Sami Płn. (Szwecja)
smn	Sami, Inari (Finlandia)
smj	Sami, Lule (Norwegia)
smk	Sami, Lule (Szwecja)
sms	Sami, Skolt (Finlandia)
san	Sanskryt
srb	Serbski (cyrylica)
srn	Serbski (cyrylica, Bośnia i Hercegowina)
srl	Serbski (łaciński)
srs	Serbski (łaciński, Bośnia i Hercegowina)
sky	Słowacki
slv	Słoweński
nso	Soto Płn.
swk	Swahili
syr	Syryjski
svf	Szwedzki (Finlandia)
sve	Szwedzki
tha	Tajlandzki
tam	Tamilski
ttt	Tatarski
tel	Telugu
tsn	Tswana
trk	Turecki
ukr	Ukraiński
urd	Urdu
uzb	Uzbecki (cyrylica)
uzb	Uzbecki (łaciński)
cym	Walijski
hun	Węgierski
vit	Wietnamski
its	Włoski (Szwajcaria)
ita	Włoski (Włochy)
zul	Zulu

5.2 Aneks B - przykładowe opisy kolumn zbiorów danych

5.2.1 Opis prosty

```
WAVE L3
NRANK L4
SEX L1
AGE L1
EDU L1
HHSIZE L1
HHBABY L2
INCOME L2
VOIVOD L2
MOTHER L1
MB_AUD L1
MB_VGA L1
MB_LAN L1
PCCORE L1
PCSPEED L2
PCMEM L1
HDBRAND L1
HDSIZE L1
CDBRAND L1
CDKIND L1
MONBRAND L1
MONSSIZE L1
NPRINTER L1
IPRINTER L1
LPRINTER L1
SCANNER L1
WEIGHT L10
```

5.2.2 Opis SPSS

```
FILE HANDLE DEMO /NAME='Demo.dat' /LRECL=45.
DATA LIST FILE=DEMO RECORD=1/
WAVE 1-3
NRANK 4-7
SEX 8
AGE 9
EDU 10
HHSIZE 11
HHBABY 12-13
INCOME 14-15
VOIVOD 16-17
MOTHER 18
MB_AUD 19
MB_VGA 20
MB_LAN 21
PCCORE 22
PCSPEED 23-24
PCMEM 25
HDBRAND 26
HDSIZE 27
CDBRAND 28
CDKIND 29
MONBRAND 30
MONSSIZE 31
NPRINTER 32
IPRINTER 33
LPRINTER 34
SCANNER 35
WEIGHT 36-45 (F)
.
```

5.3 Aneks C - zmiany

W tym temacie zostały opisane zmiany dokonane w kolejnych wersjach aplikacji.

Wersja 4.13.a wydana 2012-05-29

Zmiany tylko w programie YAC Data Analyzer.

Wersja 4.13 wydana 2011-02-03

Poprawiona obsługa instrukcji `excludeSysMis`.
Dodana instrukcja `hideRowsCols` określająca domyślną obsługę pustych wierszy i kolumn w YDA.

Wersja 4.12 wydana 2010-09-19

Dodany atrybut odpowiedzi `substStats` automatycznie przełącza obliczane wyniki w YDA z procentów na średnie.

Wersja 4.11 wydana 2010-08-15

Dodane:
- Instrukcja `fixedRecordsets` blokująca możliwość zmiany zbioru danych w analizach.
- Instrukcje `stdStats oraz advStats` pozwalające definiować grupy statystyk: standardowych i zaawansowanych.
- Instrukcja `stats` w zbiorze danych pozwalająca ograniczyć dostępne statystyki dla analiz opartych na tym zbiorze danych.
- Instrukcja `hidden` w zbiorze danych pozwalająca zablokować analizy oparte na tym zbiorze danych.

Wersja 4.10 wydana 2010-03-07

Zmiany tylko w programie YAC Data Analyzer.

Wersja 4.03 wydana 2010-01-14

Zmiany tylko w programie YAC Data Analyzer.

Wersja 4.02 wydana 2009-10-04

Dodane:
- Instrukcja `expires` w sekcji `survey` określająca datę, do której badanie jest dostępne.
- Instrukcja `minSelCount` w sekcji `waves` określająca minimalną liczbę fal potrzebnych do analizy.

Wersja 4.01 wydana 2009-07-24

Dodana kontrola licencji na pliki danych oparta na kluczach sprzętowych.

Wersja 4.00 wydana 2009-06-01

Obsługa danych telewizyjnych.

Wersja 3.04 wydana 2008-12-15

- Rejestracja powiązania rozszerzeń plików.
- Poza tym zmiany w programie YAC Data Analyzer.

Wersje 3.03.a - 3.03.d wydane między 2008-05-01 a 2008-08-24

Tylko drobne poprawki.

Wersje 3.00 - 3.03 wydane miêdzy 2007-09-29 a 2008-04-15

- Dodany prawy margines w edytorze pokazujący pozycje wszystkich uwag (błędów, ostrzeżeń oraz podpowiedzi).
- Dodane sekcje opisujące definicje prasy i radia w angielskiej wersji dokumentu.
- Można definiować skale pomiaru dla pytań.
- Skale pomiaru i atrybuty noMean mogą być automatycznie importowane z plików danych SPSS.
- Skala pomiaru wyświetlana jest (dla każdej zmiennej) w oknie dialogowym importowania.
- Więcej przykładów definicji pytań (opartych na danych General Social Survey (Generalnego Sondażu Społecznego)).
- Różne drobne poprawki.

Wersje 2.30 - 2.30.b wydane miêdzy 2006-07-31 a 2006-09-02

Zmiany tylko w programie YAC Data Analyzer.

Wersja 2.29 wydana 2006-06-04

- Zmiana systemu ochrony badań; dodatkowa definicja `licenses`.
- Dodana instrukcja `logoPlacement`.
- Dodana obsługa formatów JPEG i GIF w instrukcji `logo`.
- Dodane tłumaczenie dokumentacji YAC Data Builder na angielski.

Wersja 2.28 wydana 2005-09-26

- Darmowa dystrybucja wersji Lite z możliwością przetwarzania zbiorów o maksymalnej liczbie przypadków 1100 i maksymalnej liczbie kolumn 100.
- Dystrybucja, wraz z programem, wersji demonstracyjnej fikcyjnego badania.

Wersja 2.27 wydana 2005-08-03

- Obsługa długich nazw zmiennych (kolumn) plików SPSS (przy importowaniu danych) - obsługiwane są zmienne, których nazwy zajmują do 64 bajtów.
- Zmiana systemu licencjonowania programów YAC Data Builder i YAC Data Analyzer.
- Zmiana systemu zabezpieczania badań chronionych.

Wersja 2.26 wydana 2005-03-16

Dodatkowy pozycja w menu: **Plik | Dodaj blok...** dodaje zaznaczony tekst w edytorze do innego, istniejącego pliku (**Plik | Zapisz blok...** zastępuje tekst w innym pliku tekstem zaznaczonym w edytorze).

Wersja 2.25 wydana 2004-11-25

Dostępny jest nowy kreator: `pressReadership` (instrukcja `wizards`).

Wersja 2.24 wydana 2004-10-26

Dostępne są nowe kreatory: `cmpGroups2`, `genComplex`, `pressCmpGroups2`, `pressStructureRow` (instrukcja `wizards`).

Wersja 2.23

W tej wersji zmieniana była przede wszystkim aplikacja YAC Data Analyzer. W programie YAC Data Builder pojawiły się tylko mniejsze drobiazgi.

Wersja 2.22

Język:
- Dodany atrybut odpowiedzi `noMean` wyłączający odpowiedzi z liczenia średnich.

Wersja 2.21

Język:
- Dodana instrukcja `excludeSysMis` w sekcji `survey` ustawiająca domyślne traktowanie systemowych braków danych przy obliczeniach.

Wersja 2.20

Język:
- Definicje pytań wielowymiarowych.
- Nowy atrybut odpowiedzi: `allowOverlaps`.

Import danych:
- Automatyczne nazywanie pytań, o ile zmienne należące do pytania mają podobne etykiety.
- Kopiowanie definicji modułów i pytań jak również samych kolumn.
- Zmiana oryginalnej kolejności elementów.
- Włączanie / wyłączanie elementów do / z już wprowadzonych definicji.
- Zapis / odczyt ustawień importu.
- Podgląd definicji zmiennych (etykiety wartości, braki danych).
- Automatyczne definiowanie pytań wielowymiarowych.
- Automatycznie generowana nazwa badania uwzględnia teraz nazwę samego importowanego pliku, bez informacji o ścieżce i rozszerzeniu.
- Pół-automatyczne wybieranie zmiennych do pytania.
- Automatyczne zapisywanie ustawień importu do pliku `.dbi` przed importem danych.

Poprawki / uzupełnienia:
- Nie działała w pełni funkcja **Procesor | Uruchom YDA...** - YAC Data Analyzer nie otwierał automatycznie badania, jeżeli w nazwie pliku danych była spacja.

Wersja 2.10

Nowości:
- Integracja przetwarzania danych z importowaniem danych, w pełni funkcjonalnym edytorem i automatycznych uruchamianiem badania pod aplikacją YAC Data Analyzer.
- Instrukcja `independentColumns` w definicjach wskaźników prasowych.

Poprawki / uzupełnienia:
- Opis wcześniej nie udokumentowanego klucza odpowiedzi `attr`.

Wersja 2.00

Nowe definicje:
- Dodana instrukcja `logo` w sekcji `survey` pozwalająca na zdefiniowania logo firmy badawczej lub badania, które będzie pokazywane w programie YAC Data Analyzer w pasku stanu.
- Dodany kreator optymalizatorów media-planów prasowych (wartość `pressMediaPlanOpt` w kluczu `wizards` w sekcji `survey`).

Poprawki / uzupełnienia:
- Klucz `demo` w sekcji `survey` używany jest tylko do wyświetlania informacji w programie YAC Data Analyzer o niereprezentatywności danych. Wielkość danych nie jest kontrolowana i ograniczana.
- Program kontroluje poprawność definicji wersji w bazie danych licencji:
  - `press.media.plan` implikuje `general`,
  - `press.media.plan.opt` implikuje `press.media.plan`.

  Jeżeli definicja wersji niezgodna jest z powyższym schematem, zostanie zgłoszony błąd.
- Dodano opis nieudokumentowanego zapisu pytań multi-choice, które w pliku danych zapisane są na kilku zmiennych, w których wymienione są kody wybranych odpowiedzi (więc nie zmiennych dychotomicznych). Patrz też klucz `columnList`.

Wersja 1.20

- Uruchomiono obsługę danych prasowych (instrukcja `press`).
- Uruchomiono pełną wersję ochrony badań (licencji).
- Wprowadzono instrukcje `version` oraz `protected` do definicji `survey`.

Wersja 1.10

- Uruchomiono obsługę danych radiowych (instrukcja `radio`).
- Wprowadzono dodatkowe pole definiujące format opisu kolumn w plikach danych (`colSpecFormat`).

Wersja 1.00

- Dodano mo¿liwoœæ grupowania kodów w ramach jednej odpowiedzi (instrukcja `range`).
- Dodano odpowiedzi numeryczne (instrukcja `numeric`).

Rozdział VI

6 Okna dialogowe

6.1 Plik | Importuj...

Jest to narzędzie służące do importowania danych i opisów danych z innych aplikacji.

Obecnie dostępny jest import z formatu SPSS (plików o rozszerzeniu `.sav`).

Importowane pliki przetwarzane są na format Fixed-ASCII, które następnie przetwarzane są przez program YAC Data Builder na format używany przez aplikację YAC Data Analyzer.

6.1.1 Opcje

Zakładka ta pozwala na zdefiniowanie dodatkowych opcji importu danych.

Definiuj skale używana jest do automatycznego zdefiniowania skal pomiaru na podstawie definicji w importowanych danych.

Definiuj atrybuty noMean automatycznie definiuje ten atrybut dla braków danych (missing values) w importowanym pliku:
- jeżeli **Definiuj skale** jest zaznaczone, atrybut będzie definiowany dla braków danych w pytaniach ze skalą przedziałową,
- jeżeli **Definiuj skale** nie jest zaznaczone, atrybut będzie definiowany dla wszystkich braków danych.

Indeks

P

Q

R

S

T

W

Y

Z