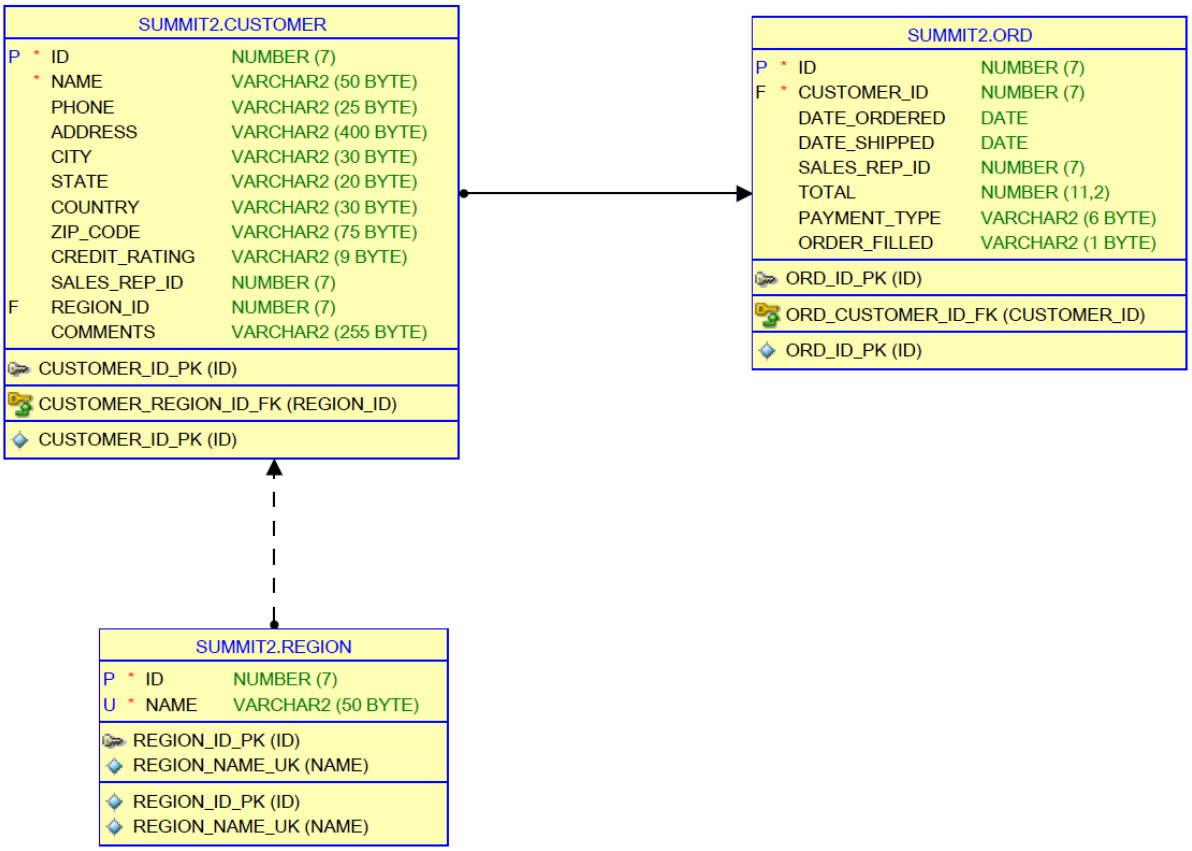
BULK INSERT (ładowanie masowe) – kolejny przykład

Opracował: dr hab. inż. Artur Gramacki

1. Model wejściowy. Uwaga: rysunek powstał w systemie ORACLE, więc nie jest on w pełni zgodny z serwerem SQLServer. Przykładowo typ `VARCHAR2` nie istnieje w serwerze SQL SERVER (nie ma dwójki na końcu). Nie ma też typu `NUMBER` (trzeba zastąpić go stosownym odpowiednikiem.



2. Tworzymy w SQL SERVER docelowe tabele wykorzystując skrypty z systemu ORACLE, z którego pobrano dane.

   - `CUSTOMER.sql`
   - `CUSTOMER_CONSTRAINT.sql`
   - `CUSTOMER_REFCONSTRAINT.sql`
   - `ORD.sql`
   - `ORD_CONSTRAINT.sql`
   - `ORD_REFCONSTRAINT.sql`
   - `REGION.sql`
   - `REGION_CONSTRAINT.sql`

   Zwróćmy uwagę, że nie wszystkie polecenia z powyższych skryptów można przenieść bezpośrednio do skryptów SQL SERVER -a. Składnia niektórych poleceń nie jest zgodna z SQL SERVER-em. Trzeba trochę samodzielnie „pokombinować" a tam gdzie trudno samemu pewnych rzeczy się domyśleć, zaglądnąć do dokumentacji serwera bazodanowego ORACLE

oraz SQL SERVER. Wynikowy skrypt pokazujemy niżej (uwaga: nawiasy kwadratowe nie są tutaj niezbędne. Potrafisz wyjaśnić dlaczego?).

```
USE [test]
GO

DROP TABLE [dbo].[ord]
GO

DROP TABLE [dbo].[customer]
GO

DROP TABLE [dbo].[region]
GO

CREATE TABLE [dbo].[customer](
      [id] [int] NOT NULL,
      [name] [varchar](50) NULL,
      [phone] [varchar](25) NULL,
      [address] [varchar](400) NULL,
      [city] [varchar](30) NULL,
      [state] [varchar](30) NULL,
      [country] [varchar](30) NULL,
      [zip_code] [varchar](75) NULL,
      [credit_rating] [varchar](9) NULL,
      [sales_rep_id] [int] NULL,
      [region_id] [int] NULL,
      [comments] [varchar](255) NOT NULL,
      CONSTRAINT [customer_PK] PRIMARY KEY ([id])
)
GO

CREATE TABLE [dbo].[ord](
      [id] [int] NOT NULL,
      [customer_id] [int] NULL,
      [date_ordered] [date] NULL,
      [date_shipped] [date] NULL,
      [sales_rep_id] [int] NULL,
      [total] [numeric](11, 2) NULL,
      [payment_type] [varchar](6) NULL,
      [ordr_filled] [varchar](1) NULL,
      CONSTRAINT [ord_PK] PRIMARY KEY ([id])
)
GO

CREATE TABLE [dbo].[region](
      [id] [int] NOT NULL,
      [name] [varchar](50) NULL,
      CONSTRAINT [PK_region] PRIMARY KEY ([id])
)
GO

ALTER TABLE [dbo].[ord] ADD CONSTRAINT [ord_customer_id_FK] FOREIGN
KEY([customer_id])
REFERENCES [dbo].[customer] ([id])
GO

ALTER TABLE [dbo].[customer] ADD CONSTRAINT [customer_region_id_FK]
FOREIGN KEY([region_id])
REFERENCES [dbo].[region] ([id])
GO
```

```
ALTER TABLE [dbo].[ord] ADD CONSTRAINT [ord_payment_type_CK]
CHECK  (([payment_type]='CREDIT' OR [payment_type]='CASH'))
GO

ALTER TABLE [dbo].[customer] ADD CONSTRAINT
[customer_credit_rating_CK]
CHECK  (([credit_rating]='POOR' OR [credit_rating]='GOOD' OR
[credit_rating]='EXCELLENT'))
GO
```

3. Za pomocą narzędzia *bcp* (jest w katalogu *c:\Program Files\Microsoft SQL Server\Client SDK\ODBC\110\Tools\Binn*) tworzymy tzw. pliki z formatem (ang. *format file*). Szczegóły tego kroku – patrz poprzednie demo.

4. Przygotowujemy polecenie ładowania masowego danych (`BULK INSERT`). Szczegóły tego kroku – patrz poprzednie demo.

5. **Ważne uwagi**: zanim uruchomimy finalne polecenie `BULK INSERT` musimy ręczne wyedytować pliki z formatem *.fmt. Będą trzy takie pliki, gdyż mamy trzy tabele, które zamierzamy zaimportować do SQL SERVER-a. Jeżeli tego nie uczynimy, to import nie uda się. Spowodowane to będzie tym, że tabele utworzone w punkcie 2 zakładają, że dane znakowe nie zawierają cudzysłowów z przodu frazy oraz z tyłu frazy. Przykładowo spójrzmy na plik `ORD_DATA_TABLE.tsv`:

```
100   204   31-08-1992  10-09-1992  11    601100      "CREDIT"    "Y"
101   205   31-08-1992  15-09-1992  14    8056.6      "CREDIT"    "Y"
102   206   01-09-1992  08-09-1992  15    8335        "CREDIT"    "Y"
103   208   02-09-1992  22-09-1992  15    377         "CASH"      "Y"
104   208   03-09-1992  23-09-1992  15    32430       "CREDIT"    "Y"
105   209   04-09-1992  18-09-1992  11    2722.24     "CREDIT"    "Y"
106   210   07-09-1992  15-09-1992  12    15634       "CREDIT"    "Y"
107   211   07-09-1992  21-09-1992  15    142171      "CREDIT"    "Y"
108   212   07-09-1992  10-09-1992  13    149570      "CREDIT"    "Y"
109   213   08-09-1992  28-09-1992  11    1020935     "CREDIT"    "Y"
110   214   09-09-1992  21-09-1992  11    1539.13     "CASH"      "Y"
111   204   09-09-1992  21-09-1992  11    2770        "CASH"      "Y"
97    201   28-08-1992  17-09-1992  12    84000       "CREDIT"    "Y"
98    202   31-08-1992  10-09-1992  14    595         "CASH"      "Y"
99    203   31-08-1992  18-09-1992  14    7707        "CREDIT"    "Y"
112   210   31-08-1992  10-09-1992  12    550         "CREDIT"    "Y"
```

oraz na definicję tabeli:

```
CREATE TABLE [dbo].[ord](
      [id] [int] NOT NULL,
      [customer_id] [int] NULL,
      [date_ordered] [date] NULL,
      [date_shipped] [date] NULL,
      [sales_rep_id] [int] NULL,
      [total] [numeric](11, 2) NULL,
      [payment_type] [varchar](6) NULL,
      [ordr_filled] [varchar](1) NULL,
      CONSTRAINT [ord_PK] PRIMARY KEY ([id])
)
```

Problem pojawi się na przykład z kolumną `payment_type`, która ma zdefiniowaną długość równą 6. A w pliku `ORD_DATA_TABLE.tsv` mamy na przykład dane `"CREDIT"`, których długość (wliczając w to cudzysłowy) wynosi aż 8. Oczywistym jest więc, że danej w tej postaci nie uda się zapisać do tabeli w SQL SERVER, mając do dyspozycji automatycznie wygenerowane narzędziem *bcp* pliki z formatami. Można oczywiście zmienić ręcznie definicję kolumny na

```
[payment_type] [varchar](8) NULL
```

ale nie jest to zbyt eleganckie rozwiązanie (dlaczego?).

Kolejny problem pojawi się np. w tym miejscu skryptu:

```
ALTER TABLE [dbo].[ord] ADD CONSTRAINT [ord_payment_type_CK]
CHECK  (([payment_type]='CREDIT' OR [payment_type]='CASH'))
GO
```

Powyższa definicja zakłada, że w kolumnie `payment_type` mogą znajdować się tylko dwie ściśle określone wartości danych. Oczywiście tutaj też problemem będą znaki cudzysłowu, które w oryginalnym pliku `tsv` są separatorami danych znakowych.

Reasumując, aby uniknąć zasygnalizowanych wyżej problemów należy PRZED uruchomieniem polecenia `BULK INSERT` dokonać stosowych ręcznych poprawek w plikach z formatami. Szczegóły tych poprawek – patrz poprzednie demo. Po tych poprawkach dopiero zawartość plików tsv będzie w 100% zgodna z definicjami tabel z punktu 2.

Można też rozważyć inne podejście do omawianego tutaj problemu i zanim rozpoczniemy importowanie danych do SQLServer-a, możemy pousuwać z plików *.tsv wszystkie ograniczające cudzysłowy. Można to zrobić, w zależności od wielkości plików, albo z wykorzystaniem dowolnego edytora tekstowego z funkcją „Find & Replace”, albo też wykorzystać jakiś bardziej efektywny sposób parsowania pliku (np. używając Linux-ego bardzo wydajnego edytora strumieniowego do przetwarzania tekstów o nazwie *sed*). Wówczas plik źródłowy dla SQLServera będzie miał postać (na przykładzie `ORD_DATA_TABLE.tsv`):

```
100   204   31-08-1992  10-09-1992  11    601100      CREDIT      Y
101   205   31-08-1992  15-09-1992  14    8056.6      CREDIT      Y
102   206   01-09-1992  08-09-1992  15    8335        CREDIT      Y
103   208   02-09-1992  22-09-1992  15    377         CASH        Y
104   208   03-09-1992  23-09-1992  15    32430       CREDIT      Y
105   209   04-09-1992  18-09-1992  11    2722.24     CREDIT      Y
106   210   07-09-1992  15-09-1992  12    15634       CREDIT      Y
107   211   07-09-1992  21-09-1992  15    142171      CREDIT      Y
108   212   07-09-1992  10-09-1992  13    149570      CREDIT      Y
109   213   08-09-1992  28-09-1992  11    1020935     CREDIT      Y
110   214   09-09-1992  21-09-1992  11    1539.13     CASH        Y
111   204   09-09-1992  21-09-1992  11    2770        CASH        Y
97    201   28-08-1992  17-09-1992  12    84000       CREDIT      Y
98    202   31-08-1992  10-09-1992  14    595         CASH        Y
99    203   31-08-1992  18-09-1992  14    7707        CREDIT      Y
112   210   31-08-1992  10-09-1992  12    550         CREDIT      Y
```