versorgungsdaten
Deutsches Institut für Medizinische Dokumentation und Information

Informationssystem Versorgungsdaten (Datentransparenz): Nutzungsordnung: Anlage 2: Regeln für den Programmcode

Nutzungsordnung: Anlage 2: Regeln für den Programmcode

Version 01-04-000

Im Geschäftsbereich des Bundesministerium für Gesundheit

Informationssystem Versorgungsdaten (Datentransparenz): Nutzungsordnung: Anlage 2: Regeln für den Programmcode

Inhalt

Anlage 2: Regeln für den Programmcode im Falle der Auftragsdatenverarbeitung

Bitte haben Sie Verständnis dafür, dass der Aufbau Ihres Programmcodes einigen formalen Vorgaben zu genügen hat. Sie erleichtern den Mitarbeiterinnen und Mitarbeitern der Datentransparenzstelle damit die Einarbeitung in Ihren Programmcode und können die zur Prüfung notwendige Zeitspanne erheblich verkürzen. Ihr Programmcode muss es den Mitarbeiterinnen und Mitarbeitern ermöglichen, Ziel und Inhalt Ihrer Analysen schnell zu erfassen und zu beurteilen. Dazu geben wir Ihnen hier die notwendigen Konventionen an die Hand. Berücksichtigen Sie in Ihren Analysen, dass im DaTraV-System sehr große Datenmengen vorgehalten und verarbeitet werden. Pro Ausgleichsjahr liegen ca. 2,1 Milliarden Datensätze vor. Um derartig große Datenmengen effizient im Mehrbenutzerbetrieb zu verarbeiten, ist ebenfalls die Einhaltung unserer Vorgaben notwendig.

1 Programmcoderegeln

1.1 Allgemeine Programmcoderegeln

- Verwenden Sie bitte nur die Sprache Deutsch oder Englisch.
- Machen Sie sich mit unserer Benennungskonventionen (s. Abschnitt 3(vertraut und beachten Sie diese bei der Ausarbeitung Ihres Skripts. Vergeben Sie sprechende, inhaltsbezogene Namen für Variablen und Tabellen.
- Machen Sie die folgenden Angaben im Programmkopf[1]:
  - Projekttitel
  - Programmversion
  - Kontaktinformation
  - Allgemeine Erläuterungen
  - Ziel der Auswertung
- Beschreiben Sie den Programmablauf möglichst systematisch. Erläutern Sie dabei ggf. auch Bezüge zu vorhergehenden und zukünftigen Auswertungen.
- Listen Sie alle verwendeten Tabellen und Variablen der DaTraV-Daten und Kataloge.
- Listen Sie ebenfalls alle von Ihrem Programmcode neu erzeugten Variablen und erläutern Sie deren Bedeutung. Stellen Sie auch den Bezug zu den Tabellen und Variablen der DaTraV-Daten und Kataloge dar.
- Unterteilen Sie Ihr Programm entsprechend dem Programmablauf in verschiedene, nummerierte Abschnitte mit aussagekräftigen Titeln. Achten Sie dabei darauf den Programmcode optisch klar zu strukturieren.
- Erläutern Sie für jeden Abschnitt dessen Zweck im Kontext des von Ihnen beschriebenen Programmablaufs.
- Wenn Sie zu jedem Abschnitt zusätzlich und hinreichend konkret beschreiben welches Ergebnis Sie erwarten, können wir prüfen, ob Ihre Erwartung erfüllt wird und Sie, sofern Sie dies explizit wünschen, informieren, wenn dies nicht der Fall ist.
- Erläutern Sie ggf. auch einzelne Anweisungen und deren Komponenten.
- Bitte berücksichtigen Sie, dass Ausgaben von Daten / Ergebnissen, die über die Konsole erfolgen, nicht an Sie übermittelt werden.

[1] Gerne können Sie als Vorlage den Programmkopf aus dem Beispielskript auf unserer Webseite verwenden.

Informationssystem Versorgungsdaten (Datentransparenz): Nutzungsordnung: Anlage 2: Regeln für den Programmcode

- Benennen Sie alle Tabellen nach einem einheitlichen Schema, welches u.a. ein Kürzel für Ihre Institution oder das Projekt vorsieht. Beachten Sie, dass nur Ergebnistabellen von uns exportiert werden und deren Namen das Präfix ‚RT_' enthalten muss.
- Tabellen müssen selbsterklärend sein, d.h. aus den Tabellenspalten und -zeilen muss der Tabelleninhalt eindeutig hervorgehen[2]. Tabellen müssen darüber hinaus handhabbar sein.
- Erstellen Sie keine Kopien der Originaldaten. Dies kostet unnötig Speicherplatz, reduziert die Ausführungsgeschwindigkeit Ihres Skriptes und stellt ein Sicherheitsrisiko dar. Arbeiten Sie mit frühzeitiger Datenreduktion über Filterbedingungen, wo immer möglich.
- Vermeiden Sie tiefe Schachtelung von Verzweigungen und / oder Schleifen.
- Vermeiden Sie die Verkettung mehrerer Unterabfragen. Erzeugen Sie stattdessen mehrere temporäre Zwischentabellen, die Sie anschließend geeignet zusammenführen.
- Achten Sie darauf, dass Abfragen und Unterabfragen nicht wiederholt erstellt werden. Statt diese wiederholt zu erstellen, sollten Sie eine temporäre Tabelle mit dem Ergebnis dieser Abfrage erzeugen und diese dann, wo immer erforderlich, einbinden.
- Verzichten Sie auf ‚SELECT *' abfragen. Benennen Sie in einem SELECT-Statement die einzubeziehenden Spalten.
- Bitte setzen Sie Pfadreferenzen auf verwendete Dateien an den Programmanfang.
- Verwenden Sie maximal 132 Zeichen pro Zeile.
- Rücken Sie Schleifen bitte einheitlich ein und fügen Sie Abstände zwischen den Programmblöcken ein.
- Die Groß- und Kleinschreibung sollte einheitlich verwendet werden.
- Bitte wechseln Sie nicht die Kennzeichnungen für im Code benutzte Objekte (Beispiel: Vergleichsoperatoren [entweder einheitlich „<>" oder „!="]).
- Achten Sie auf ein angemessenes Laufzeitverhalten und Fehlerfreiheit. Testen Sie dazu Ihren Code syntaktisch mit Hilfe des Beispieldatensatzes.
- Sorgen Sie dafür, dass Ihr Programm in einem Schritt ausführbar ist. Nennen Sie die auszuführende Hauptdatei, wenn Sie mit Zusatzdateien arbeiten. Nummerieren Sie Dateien im Dateinamen gemäß der Ausführungsreihenfolge.
- Programme müssen wiederholt fehlerfrei ausführbar sein. Bauen Sie ggf. entsprechende Bereinigungsfunktionen in Ihre Programme ein.

[2] Dies ist bei den zur Verfügung stehenden 30 Zeichen nicht immer einfach oder möglich. Es kann hilfreich sein ausgehend von unseren Benennungskonventionen eigene Benennungsschemata festzulegen.

- Bevor eine Tabelle erstellt wird, muss sichergestellt werden, dass eine Tabelle gleichen Namens noch nicht oder nicht mehr existiert. Dazu sollten Sie folgende Oracle-Prozedur definieren:

```
CREATE OR REPLACE PROCEDURE DeleteIfExists (
   lcObjName VARCHAR2,
   lcObjType VARCHAR2 DEFAULT 'Table'
) AUTHID CURRENT_USER
IS
   lnCount NUMBER:=0;
BEGIN
   CASE
      WHEN UPPER(TRIM(lcObjType)) = 'TABLE' THEN
         SELECT COUNT(*) INTO lnCount FROM user_tables
         WHERE table_name = UPPER(TRIM(lcObjName));
         IF lnCount > 0 THEN
           EXECUTE IMMEDIATE 'DROP TABLE'
           || TRIM (lcObjName) || ' CASCADE CONSTRAINTS';
         END IF;
      WHEN UPPER(TRIM(lcObjType)) = 'VIEW' THEN
         SELECT COUNT(*) INTO lnCount FROM tab
          WHERE tname = UPPER(TRIM(lcObjName));
          IF lnCount > 0 THEN
            EXECUTE IMMEDIATE 'DROP VIEW’
            || TRIM (lcObjName);
          END IF;
   END CASE;
END;
/

EXECUTE DeleteIfExists(‘meineTabelle‘)
```

- Zusatzdaten sind nur erlaubt, wenn es sich um kleine Werte- bzw. Schlüsseltabellen mit einer überschaubaren Anzahl von Einträgen handelt.
- Die Verwendung von zusätzlichen Bibliotheken, insbesondere Bibliotheken von Drittanbietern, ist untersagt. Halten Sie den Code der eingereichten Skripte möglichst kurz und einfach um den Prüfaufwand zu reduzieren.
- Verwenden Sie keine DML-Befehle wie INSERT, UPDATE, DELETE und MERGE da diese auf großen Datenmengen lange Laufzeiten und einen hohem Ressourcenverbrauch verursachen. Benutzen Sie stattdessen „CREATE TABLE AS SELECT“ und „UNION ALL“. Vermeiden Sie die Erstellung von Tabellen per CREATE TABLE, die nachträglich über DML-Befehle (z.B. DELETE FROM oder ALTER TABLE) minimiert werden, da dies sehr ressourcenintensiv ist
- Filtern Sie die Daten frühzeitig über CREATE-TABLE Statements mit WHERE-Bedingungen, da sich dies positiv auf die Ausführungsgeschwindigkeit Ihres Skriptes auswirkt.

1.2 Programmcoderegeln mit speziellem Fokus auf den Identifikationsschutz

- Ergebnistabellen, die außerhalb der Datenaufbereitungsstelle verwendet werden sollen, müssen auf Wahrung der Anonymität überprüft werden. Neben den zuvor genannten Hinweisen zum Aufbau des Programmcodes sollten daher die folgenden Punkte berücksichtigt werden:
  - Für die Prüfung wird die Information benötigt, auf Basis welcher Fallzahlen die einzelnen Werte in einer Tabelle berechnet worden sind. Dies gilt beispielsweise für die Berechnung von Anteilswerten. Bei der Verwendung von Gewichtungsfaktoren müssen auch die ungewichteten Fallzahlen abgebildet werden.
  - Geringe Fallzahlen (ungewichtet unter X) sollten Sie durch geeignete Maßnahmen, z.B. Bildung von Klassen oder Zusammenfassungen von Merkmalsausprägungen, möglichst vermeiden.

Informationssystem Versorgungsdaten (Datentransparenz): Nutzungsordnung: Anlage 2: Regeln für den Programmcode

- Tabellen sind mit mindestens einer Spalte für die Anzahl der distinkt gezählten Versicherten, die den aggregierten Angaben in den anderen Spalten zugrunde liegen, zu versehen. Als Spaltenname sollte CNT_D_PSID verwendet werden. Nach einem weiteren Unterstrich kann der Spaltenname um weitere alphanumerische Zeichen ergänzt werden, um beispielsweise mehrere dieser Spalten in einer Tabelle voneinander unterscheiden zu können und Zuordnung zu den Spalten mit den aggregierten Angaben zu ermöglichen. Die CNT_D_PSID-Spalten werden für die Umsetzung der Mindestfallzahlregelung benötigt.
- Bedenken Sie die tabellenübergreifende Geheimhaltung. Vermeiden Sie z.B. bei der Verwendung von Filterbedingungen Restmengen, die sich als Differenz zu anderen dargestellten Mengen erschließen lassen. Beispiel: Sie erzeugen eine „Insgesamt“-Tabelle und eine Tabelle „Männlich“; die Differenz ist „Weiblich“. Erzeugen Sie in einem solchen Fall die Tabelle „Weiblich“ explizit mit, da diese bei der Prüfung der Wahrung der Anonymität berücksichtigt werden muss.
- Versuchen Sie inhaltliche Bezüge/Abhängigkeiten zwischen und innerhalb von Tabellen zu minimieren oder aufzulösen, die durch verschiedenartige Gruppierungen auf den gleichen Daten entstehen. Beachten Sie dabei, dass die Ausweisung von Werten, die Rückrechnung von modifizierten Werten ermöglichen und somit zur Deanonymisierung geeignet sind, in Ergebnistabellen nicht erlaubt sind. Nicht modifizierte Randsummen sind in jedem Fall unzulässig, wenn zugleich die den Randsummen zugrunde liegenden Werte ausgewiesen werden. Entscheiden Sie was Ihnen wichtiger ist: Die Darstellung von Randsummen oder die zugrunde liegenden modifizierten Einzelwerte.
- Vermeiden Sie Gruppierungskaskaden und inhaltliche Wiederholungen in den Ergebnistabellen.
    - Tabelle 1 – Daten A gruppiert nach Alter
    - Tabelle 2 – Daten A gruppiert nach Geschlecht
    - Tabelle 3 – Daten A gruppiert nach Alter und Geschlecht
- Sie können auch selbst die Anwendung der Mindestfallzahlregel in Ihrem eingereichten Auswertungsskript vorsehen oder andere Verfahren zur Sicherstellung der Geheimhaltung anwenden, die uns dann zu erläutern wären.
- Minima und Maxima werden im Allgemeinen nicht herausgegeben: Umgehen Sie dies z.B. durch Perzentile. Beachten Sie dabei, dass 1% und 99% mindestens eine Basis von N=300 Fälle, 5% und 95% mindestens N=60 Fälle (ungewichtet) erfordern.
- Bei Kostenstatistiken muss der größte und zweitgrößte Wert zu Kontrollzwecken ausgewiesen werden.
- Beachten Sie, dass bei der Prüfung der Wahrung der Anonymität auch die Daten, die Sie bereits erhalten haben, berücksichtigt werden. Zum einen kann dies den Informationswert der von Ihnen erstellten Ergebnisse einschränken, da sich hieraus zusätzliche Sperrungen ergeben. Zum anderen steigt damit der Aufwand der Prüfung. Beantragen Sie daher nur die Daten / Ergebnisse, die Sie wirklich benötigen.
- Erstellen Sie so wenige Tabellen mit so wenig Spalten und Zeilen wie möglich.
- Gibt es Tabellen die nur der Plausibilisierung dienen und die z.B. unmodifizierte Randsummen enthalten, dann trennen Sie diese Tabellen namentlich per Präfix „CK_“ von den eigentlichen Ergebnistabellen. Tabellen die Plausibilisierungszwecken dienen sollten nicht Teil der Ergebnismenge sein. Nach Ihren Vorgaben kann dann die Plausibilisierung durch uns vorgenommen werden. Sollte dabei Widersprüchliches oder Überraschendes zutage treten, kann ggf. ein Besuch am vorläufigen Gastarbeitsplatz vereinbart werden.
- Vermeiden Sie einen unnötigen Detaillierungsgrad. Wenn Sie kalkulierende SQL-Funktionen verwenden runden Sie die Ergebnisse auf eine sinnvolle Anzahl von Nachkommastellen.

1.3 Programmcoderegeln für den Datenbankzugriff (Oracle)

- Fügen Sie in Ihren Programmcode auskommentierte Blöcke ein, um …
  - alle temporären Tabellen gezielt löschen zu können.
  - alle Ergebnistabellen gezielt löschen zu können.
  - um alle anderen Tabellen gezielt löschen zu können, sofern Sie zwischen mehr als den beiden zuvor genannten Tabellentypen unterscheiden.
- Benutzen Sie für das Ansprechen der von Ihnen erzeugten Tabellen keine Schema -Objekt Präfixe wie mein_schema.meine_tabelle
- Fügen Sie einzelne Ergebnistabellen nur dann zu einer Ergebnistabelle zusammen, wenn sich dadurch der Tabelleninhalt und der im Grunde gegebene Bezug zwischen den einzelnen Ergebnistabellen leichter erschließen lässt. Vermeiden Sie das Zusammenfügen von sehr großen Tabellen.
- Verwenden Sie bei CREATE TABLE Anweisungen die Optionen COMPRESS und NOLOGGING.
- Verwenden Sie keine ORDER BY Anweisungen in CREATE Table Anweisungen. Diese werden bei der Erstellung einer Tabelle ignoriert. Verwenden Sie Sortierungen nur, wenn dies nötig ist. Insbesondere die Sortierung großer Datenmengen wirkt sich negativ auf die Laufzeiten aus.
- Verwenden Sie die ANSI-92 Syntax mit explizit formulierten JOIN- Bedingungen nicht die ältere unübersichtlichere ANSI-89 Syntax.
- Verwenden Sie nicht den Oracle Plusoperator (+).
- Vermeiden Sie umfangreiche CASE WHEN Abfragen zum Beispiel zur Altersgruppen-Bestimmung. Verwenden Sie in diesen Fällen lieber sogenannte Schlüsseltabellen, die Sie dann über einen JOIN einbinden.

1.4 Programmcoderegeln für SAS

- Reichen Sie Skripte bitte als UTF-8 Textdateien ein.
- Liefern Sie Ihre SAS-Skripte in einem Format, das in der Version 9.4 lesbar ist.
- Vermeiden Sie die Erstellung von Grafiken als Teil einer Ergebnismenge. Nutzen Sie, wo immer möglich, Tabellen an Stelle von Grafiken.
- Verwenden Sie nur die Basismodule BASE, CORE, STAT.
- Nutzen Sie die ORACLE Datenbank zur Speicherung Ihrer Zwischenergebnisse und Ergebnisse. Verwenden Sie hierzu SQL Passthrough (s. folgenden Abschnitt). Beachten Sie hierbei bitte alle zuvor aufgeführten Programmcoderegeln.

2 Hinweise zum Arbeiten mit SAS

Die DaTraV-Rohdaten werden in der Oracle Datenbank und nicht in nicht in SAS vorgehalten. Alle zu erstellenden Tabellen sind explizit in Oracle zu speichern. Dies spart Speicherplatz und erhöht i.d.R. die Ausführungsgeschwindigkeit Ihres Skriptes. Zusätzlich ist der Datentransfer zwischen Oracle und SAS auf das nötige zu beschränken. Folgende Hinweise beschreiben, wie Sie auf die Daten zugreifen und eigene Tabellen erstellen können.

2.1 Vorgelagerte Datenaufbereitung per SQL-Skript

Der erste Schritt einer Analyse ist die Zusammenstellung der Versicherten, die in Ihre Analyse Eingang finden sollen, zu einer Auswertungspopulation. In der Regel führt dies zu einem gezielten Ausschluss von Versicherten und damit zu einer Verringerung der zu analysierenden Daten. In weiteren Schritten werden dann i.d.R. auf Basis der Auswertungspopulation weitere Versichertengruppen gebildet, die nachfolgend detailliert analysiert werden. Auch diese Schritte zur Bildung der Untersuchungsgruppen führen oft zu einer Verringerung der zu analysierenden Daten. Daher bitten wir Sie alle zumindest datenreduzierenden Schritte so früh als möglich und mit Hilfe eines, ggf. auch mehrerer SQL-Skripte zu realisieren. Optimal wäre es, wenn das SAS-Skript lediglich die Operationen ausführen würde, die nur in SAS mit vertretbarem Aufwand umsetzbar sind. Ein Vorteil dieser Herangehensweise ist eine performantere Analyse. Zudem kann so weitestgehend auf SQL-Passthrough-Befehle in SAS verzichtet werden.

2.2 Zugriff auf die Oracle Datenbank aus SAS über SQL-Passthrough

Nur die notwendigen Selektionen oder Zusammenführungen sind in SAS per SQL-Passthrough direkt auf dem ORACLE Datenbanksystem durchzuführen. Hierzu ist eine CONNECT Anweisung innerhalb eines jeden PROC-SQL-Blockes erforderlich:

```
CONNECT TO ORACLE AS con1(PATH=&ORA_SID USER="&ORA_USER" PW="&ORA_PASSWORD");
```

Die Verbindung (hier benannt als con1) wird mit den in Klammern angegebenen Zugangsdaten aufgebaut. Anschließend kann sie verwendet werden und muss am Ende des PROC-SQL-Blocks mit dem Befehl

```
DISCONNECT FROM con1;
```

wieder geschlossen werden.

Mittels eines EXECUTE-Blocks kann nun ein SQL-Statement auf der Datenbank ausgeführt werden.

Beispiel:

```
PROC SQL;
    CONNECT TO ORACLE AS con1
(PATH=&ORA_SID USER="&ORA_USER" PW="&ORA_PASSWORD");

   EXECUTE ( * siehe hierzu Abschnitt 4.6
      EXEC DeleteIfExists('RT_01_SA651_BJ');
    ) BY con1;

    EXECUTE (
      CREATE TABLE RT_01_SA651_BJ
                as SELECT DISTINCT(SA651_BERICHTSJAHR) AS D_BJ
                FROM "&DATRAV_SCHEMA".V2010SA651
    ) BY con1;

    DISCONNECT FROM con1;
quit;
```

Wird über EXECUTE ein CREATE TABLE oder ein anderer tabellen-erzeugender Befehl ausgeführt, wird die Tabelle innerhalb der Datenbank angelegt. Die Daten verbleiben somit in der ORACLE Umgebung.

2.3 Verwendung von Makrovariablen

Für den Zugriff auf die Daten der Oracle-Datenbank werden folgende Zugangsdaten benötigt:

- ORA_SID
- ORA_USER
- ORA_PASSWORD

Innerhalb jeden EXECUTE Blocks, in dem die DaTraV-Daten verwendet werden, ist zudem das Schema in dem diese Daten zu finden sind, anzugeben:

- DATRAV_SCHEMA

Insbesondere bei den Zugangsdaten handelt es sich um sensible Daten, die erst zur Laufzeit des Skripts als Parameter übergeben werden sollten. Hierzu eignen sich Makrovariablen, die durch ein vorangestelltes „&“ gekennzeichnet werden.

Bitte definieren Sie daher folgenden SAS-Makrovariablen am Anfang Ihres Skriptes:

- &ORA_SID
- &ORA_USER
- &ORA_PASSWORD
- &DATRAV_SCHEMA

Verwenden Sie dazu folgenden Befehl:

```
%LET Macro_Variable_Name = Wert;
```

Beispiel:

```
%LET ORA_PASSWORD = TopSecret111;
```

2.4 Ausgabe von Prozeduren in die Datenbank schreiben

Um Ausgaben einer Prozedur in die Datenbank schreiben zu können, müssen Sie zuvor mit folgendem LIBNAME-Statement eine SAS Bibliothek anlegen:

LIBNAME OraLib ORACLE PATH=&ORA_SID USER="&ORA_USER"
PASSWORD="&ORA_PASSWORD";

Dieses Statement sollten Sie am Beginn Ihres Skripts einfügen.

Nun können Sie die Option OUT in Ihren Anweisungen verwenden.

Beispiel für PROC FREQ:

```
/*Bibliothek in der Datenbank anlegen*/
LIBNAME OraLib ORACLE PATH=&ORA_SID USER="&ORA_USER" PASSWORD="&ORA_PASSWORD";

/*Ergebnisse in der Datenbank als Tabelle speichern*/

PROC FREQ DATA=OraLib.TT_temporäreTabelle;
       TABLE Spalte1*Spalte2 / OUT=OraLib.RT_Tabelle sparse;
       WHERE X=1;
RUN;
```

3 Benennungskonventionen

3.1 Regeln zur Benennung von Variablen

Regel	Kommentar
Die Namen lokaler Variablen haben mit ‚l' ergänzt um ein Kürzel für den Variablentyp zu beginnen: • String, Character (lc) • Number, Integer (ln, li) • Boolean (ll) • Date (ld) • Time, Time & Date Timestamp (lt) • Record (lr)	
Die Namen von Variablen, die Typdefinitionen enthalten, haben mit ‚t' oder ‚td' zu beginnen.	
Die Namen von Variablen, die auf Cursor verweisen, haben mit ‚c' zu beginnen.	
Für Variablen, die zur Steuerung des Ablaufs einer Analyse verwendet werden und ansonsten keine inhaltliche Bedeutung für die Analyse haben, sind englische Bezeichnungen zu wählen.	Bspl.: lnIndex, lnCounter
Für Variablen, die eine inhaltliche Bedeutung für die Analyse haben, sind deutsche Bezeichnungen zu wählen.	Bspl.: lnAlter, lcGeschlecht

3.2 Regeln zur Benennung von Tabellen

Kürzel	Bedeutung	Kommentar	Farbe in Grafiken
_AP	Action Protocol Table	In dieser Tabelle sind Laufzeiten und alle relevanten Aktionen zu protokollieren.	i.d.R. nicht darzustellen
_DD	Data Description Table	In dieser Tabelle sind die RTs aufzulisten. Zu jeder RT gehört eine kurze Beschreibung und ggf. ein ergänzender Kommentar.	i.d.R. nicht darzustellen
RT_	Result Table	Diese Tabellen stellen die Ergebnismenge dar.	orange
TT_	Temporary Table	Diese Tabellen können nach einem Analyselauf gelöscht werden.	weiß
VT_	Value Table		blau
IT_	Intermediat Result Table	Diese Tabellen enthalten wichtige Zwischenergebnisse.	gelb
PT_	Data Preparation Table	Diese Tabellen enthalten für bestimmte Zwecke aufbereitete DaTraV-Daten.	weiß oder dunkles grün
CK	Check Table	Diese Tabelle enthält Prüfungen und Plausibilisierungen	helles grün
BP_	Basic Population Table	Standard-Tabellen zur Definition einer Auswertungspopulation. Im Prinzip handelt es sich um spezifische PT.	dunkles grün

4 Referenzen

“Regeln für den Programmcode beim Zugangsweg kontrollierte Datenfernverarbeitung” der Statistischen Landesämter und des Statistischen Bundesamts