HIT-Batch-Schulungsunterlagen

2024

HIT/ZID-Datenbank
im Bayerisches Staatsministerium für
Ernährung, Landwirtschaft und Forsten
Ludwigstr. 2
80539 München
Stand: 11.03.2024

Inhaltsverzeichnis

Inhaltsverzeichnis 2
1. Allgemeines 3
1.1. HIT-Testsystem 3
1.2. HIT-Kompetenzen 4
2. HIT-Kommunikation 5
2.1. Unit-Test - Technischen Voraussetzungen und Installation 5
2.1.1. Geeigneten Texteditor installieren 5
2.1.2. HitUnit – Programm Download, Installation & Konfiguration 5
2.2. HIT-Protokoll 7
2.2.1. HIT-Spezifikation 7
2.2.2. HIT-Abfragen in HIT-QL (Query-Language) 12
2.2.3. HIT-Abfragefunktionen 14
2.2.4. HITP-Beispiele 17
3. HIT-Batch 18
3.1. Technische Voraussetzungen 18
3.2. Alternativ: Vereinfachte Version HIT-Batch 21
3.3. HIT-Batch-Abfragen 22
3.3.1. HITP-Vorgehen: Senden mehrerer gleicher Meldungen 23
3.3.2. HITP-Vorgehen: Abfragen mit Query-Language (HIT-QL) 23
3.3.3. Filterfunktionen 24
3.3.4. Verschiedenes 26
4. Befehlserstellung - Hilfe über Meldeprogramm 27

1. Allgemeines

1.1. HIT-Testsystem

Für Übungszwecke steht das HIT-Testsystem zur Verfügung.

- Echtdaten grundsätzlich nicht ins Testsystem!
- Testeingaben grundsätzlich nicht im Echtsystem!

So ist das HIT-Testsystem zu finden:

- Link zu HitTest auf Startseite http://www.hi-tier.de
- oder direkt http://www.hi-tier.de/HitTest/



Im HitTest-Hilfetext (Klick auf HELP?) sind Test-Zugänge beschrieben.

Bei Abfragen/Eingaben immer die Kompetenzen des Betriebstyps beachten!

Testbetriebe am Beispiel von Land 01 (Schleswig-Holstein)

Betriebsnummer	PIN	Betriebstyp
01 000 000 0001	900001	Landwirt
01 000 000 0002	900002	sonstiger Tierhalter (z.B. Zuchtverband)
01 000 000 0003	900003	Viehhändler
01 000 000 0004	900004	Schlachtbetrieb
01 000 000 0005	900005	Transporteur

Spezielle Benutzergruppen

Anmeldung als Landwirtschaftsverwaltung

01 000 000 0006	900006	Landwirtschaftsverwaltung
01 000 000 0007	900007	Landwirtschaftsverwaltung (erweiterte Rechte)

Anmeldung als Veterinärverwaltung

01 000 000 0008	900008	Veterinärverwaltung

Anmeldung als BSE-Labor

01 000 000 0081	900081	BSE-Labor
01 000 000 0082	900082	Landeslabor
01 000 000 0084	900084	amtlicher Tierarzt (BSE)

Ohrmarken zum Testen

DE 03 000 00001	DE 09 810 00000
DE 03 000 00002	DE 09 810 00001
DE 03 000 00003	DE 09 810 00002
DE 03 000 00004	DE 09 810 00003

Generell ist der Zugangsregel zum HIT-Testsystem so:

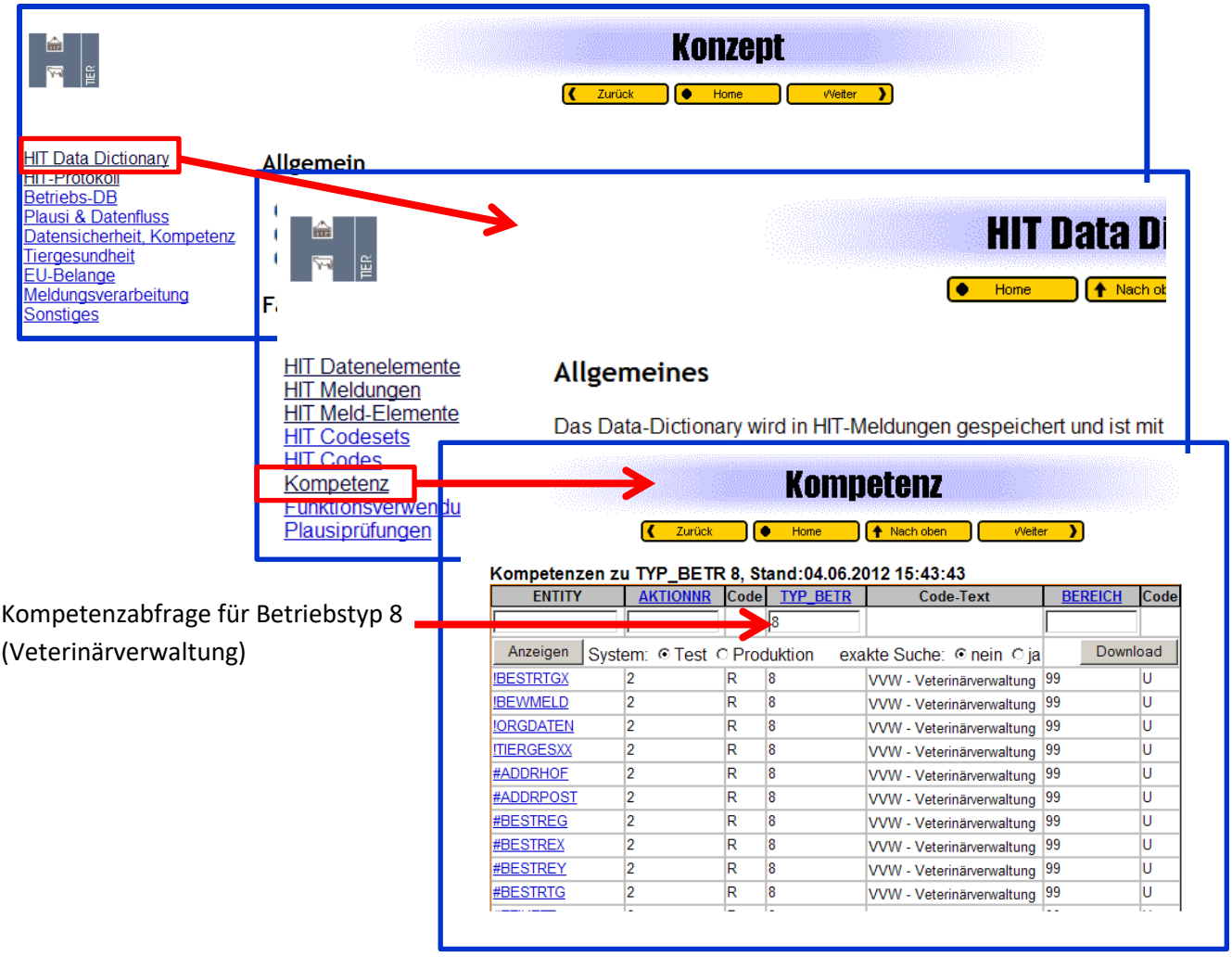
Betriebsnummer / PIN : YY000000XXXX / Aaaa$90XXXX

=> YY Bundesland z.B. 09 für BY, XXXX Betriebstyp mit dem man sich anmelden will.

z.B. Veterinärverwaltung BY: 090000000008 / Aaaa$900008

1.2. HIT-Kompetenzen

Die Betriebstypen in HIT haben unterschiedliche Kompetenzen. Diese sind Online einsehbar und bei Abfragen zu berücksichtigen, siehe Entwicklungsbereich: http://www.hi-tier.de/Entwicklung/



Kompetenzabfrage für Betriebstyp 8 (Veterinärverwaltung)

2. HIT-Kommunikation

Die Kommunikation zwischen dem Benutzer und dem HIT-System basiert auf einer Anfrage-Antwort Abfolge zwischen einem „Client-Programm“ und einem „HIT-Server“ die dabei verwendete Schreibweise oder „Sprache“ ist das HIT-Protokoll. Ein einfaches Client-Programm um das sofort aus zu probieren ist der Unit-Test.

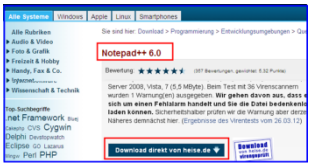
2.1. Unit-Test - Technischen Voraussetzungen und Installation

2.1.1. Geeigneten Texteditor installieren

z.B. „Notepad++6.0“ von Heise

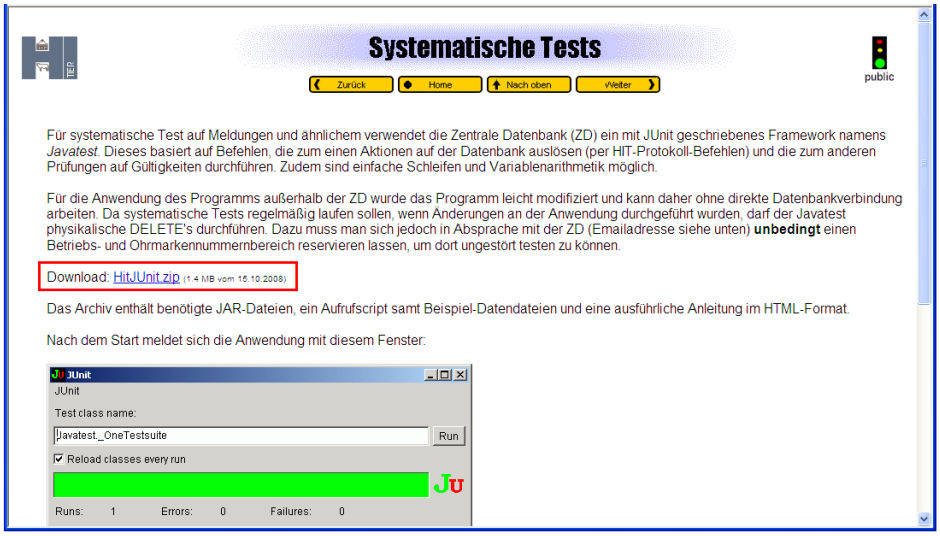
- http://www.heise.de/download/notepad.html
- Notepad auf Laufwerk speichern, auf das man Zugriff hat, installieren

Das Windows-Notepad ist wenig geeignet!



2.1.2. HitUnit – Programm Download, Installation & Konfiguration

Anwendung: Testszenarien für systematische Tests erstellen (melden und abfragen)
http://www.hi-tier.de/Entwicklung/Programme/javatest.html
„HitJUnit.zip“ auf Laufwerk speichern, auf das man Zugriff hat



2.1.2.1. Testcase.ini (Steuerdatei) anpassen

- Testcase.ini in Texteditor öffnen.
- Server-, Port-Eintrag und Testdatei überprüfen/anpassen
  - Bei BY Behördennetz (also aus bayerischem Behörden-Intranet) „… .hi-tier.bybn.de“ ansonsten „hi-tier.de“
  - „PrimaryServer“ + „BackupServer“ verschiedene Server, alternativer Zugriff
  - Port 2223 für Testsystem, Port 2222 für Produktion

Lokale Netzwerke (Behörde), die neben den üblichen Internetdiensten (HTTP, FTP) keinen direkten Zugriff auf das Internet erlauben, müssen die Ports für Test und Produktion **freischalten (lassen)**,

damit Programme wie der UnitTest oder HitBatch eine direkte Verbindung zu den HitServern aufbauen können.

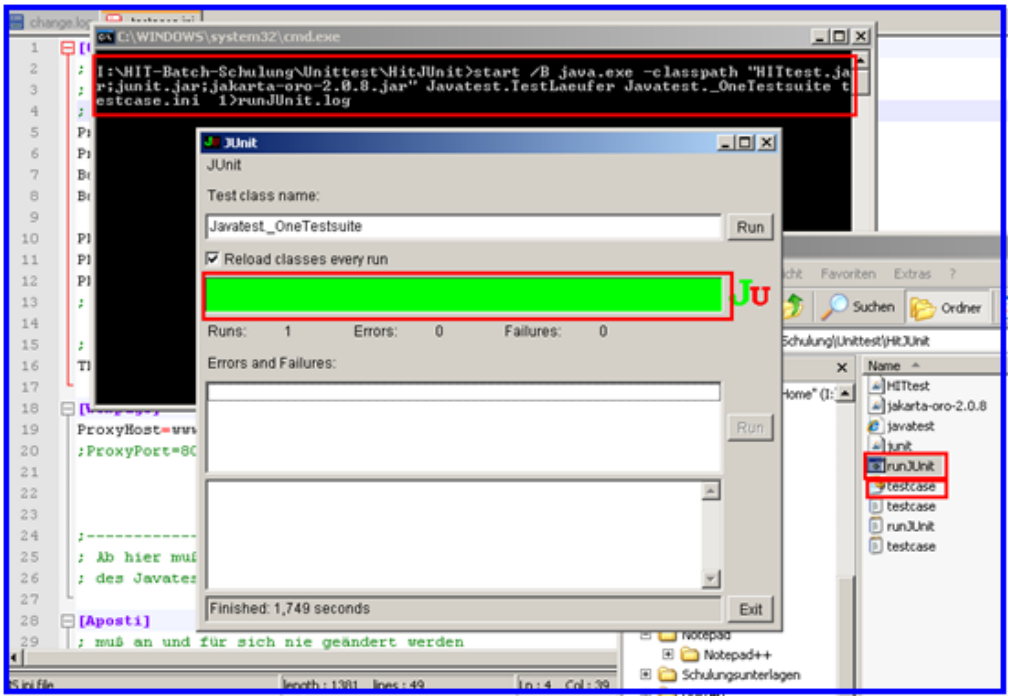
```
[Global]
; für Tests auf Test-System    die Ports 2223 oder 2226
;           auf Wartungssystem die Ports 2224 oder 2230 oder 2231
;           auf Prod-System Ports 2222
PrimaryServer=hitserver.hi-tier.bybn.de
PrimaryPort=2223
BackupServer=hitbackup.hi-tier.bybn.de
BackupPort=2223

PROTOFILE=testcase.log
PROTOAPPEND=0
PROTOLEVEL=1
; Protokollmenge wird im TESTFILE mit #ECHO gesteuert!

; Liste mit Namen der Testdatei oder Dateien mit Semikolon getrennt  (Default: leer = Javatest\_OneTest.txt)
TESTFILES=testcase.txt

[Webpage]
ProxyHost=www-proxy.bybn.de
;ProxyPort=80
```

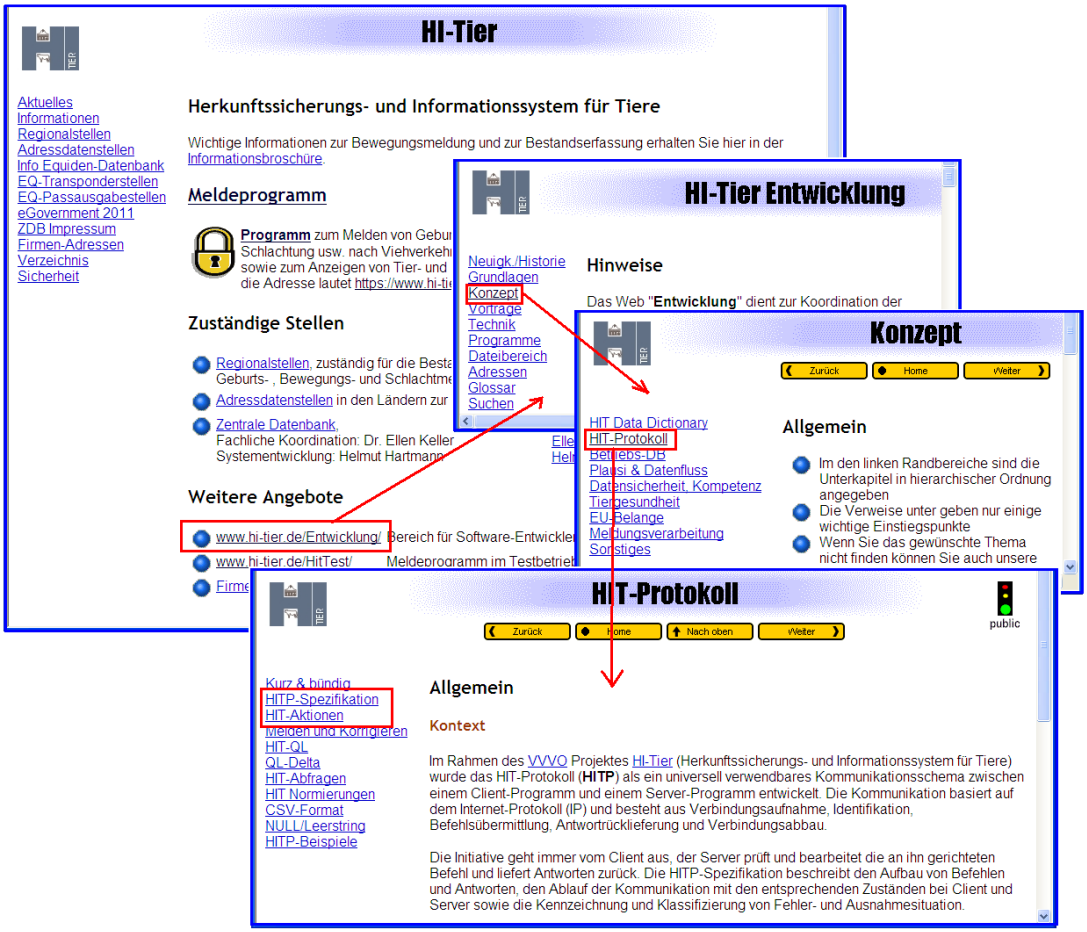
- Nach Anpassung „runJunit.cmd“ starten (Doppelklick)
- Test läuft
- grüner Balken = OK / roter Balken „testcase.log“ ansehen zur Problemprüfung



Grün bedeutet, dass das Programm entsprechend der Erwartungen gelaufen ist. Es bedeutet nicht zwingend, dass die HIT-Befehle alle „gut gegangen sind“. Wenn z.B. meine Erwartungs ist, beim Senden einer Geburtsmeldung ohne Angabe der Mutter einen Fehler zu erhalten ist das tatsächliche Auftreten dieses Fehlers auch ein „Erfolg“.

2.2. HIT-Protokoll

Das HIT-Protokoll dient zur Kommunikation zwischen externen Programmen (Clients) mit der Zentralen Datenbank (HIT-Server). Hauptzweck des Protokoll ist das Senden und Abrufen von Meldungen mit Datenelementen, analog zu Datensätze mit Datenfeldern.
Die Clients erteilen Befehle, der Server gibt Antworten.



Oder direkt über den Link: http://www.hi-tier.de/Entwicklung/Konzept/Default.htm

Die **HITP-Spezifikation** beschreibt:

- den Aufbau von Befehlen und Antworten
- den Ablauf der Kommunikation mit den entsprechenden Zuständen bei Client und Server
- die Kennzeichnung und Klassifizierung von Fehler- und Ausnahmesituation.

Unter **HIT-Aktionen** ist die Klassifizierung der Aktionen (der Teil des Befehls, der angibt was getan werden soll).
Die abstrakte Beschreibung der HIT-Query-Languange (HIT-QL) als SQL-ähnliche Abfragesprache des HIT-Protokolls, wird in einem eigenem Kapitel behandelt, siehe **HIT-QL**.
Beispiel-Abfragen siehe **HIT-Abfragen**.

2.2.1. HIT-Spezifikation

http://www.hi-tier.de/entwicklung/Konzept/HITP/feink040.html

2.2.1.1. Befehle

gliedern sich in 4 Gruppen und dienen zum:

Gruppe	Befehl	Erläuterung
1	E**x**ecute	Ausführen von Kommandos, wie z.B. Logon, Logoff, Parameter setzen sowie Ausführen von Meldungen bei denen der Sender nicht weiß, ob Insert oder Update nötig, insbesondere bei Betriebsdaten-Meldungen mit automatischer Historisierung
2	**I**nsert	Übertragen oder Pflegen von Datensätzen der einzelnen Meldungsarten
	Update	
	Storno	
	Delete	
3	**R**etrieve	Abholen von verschiedenen Daten vom Server
4	**C**onfirm	Bestätigen bereits abgespeicherter Sätze insbesondere im Konfliktfall (früher XS/S)

Befehlsaufbau

- Ein Befehl besitzt eine fortlaufende Befehls-Nummer.
- Der 1.Befehl der Connection hat die Nummer 1.
- Ein Befehl kann aus einem oder mehreren Teilbefehlen bestehen.
- Jeder Teilbefehl ist eine eigene Zeile.
- Wenn weitere Teilbefehle folgen ist die Zeile mit dem Befehlskennzeichen "+" zu beginnen, beim letzten Befehlsteil oder wenn der Befehl nur aus einem Teil besteht ist die Zeile mit dem Befehlskennzeichen "*" zu beginnen.
- Jeder Teilbefehl hat die gleiche Befehls-Nummer.
- Teilbefehle werden ohne dazwischen liegende Antworten versendet.
- Befehlsteile dienen zur blockorientierten Übertragung

Schematische Darstellung

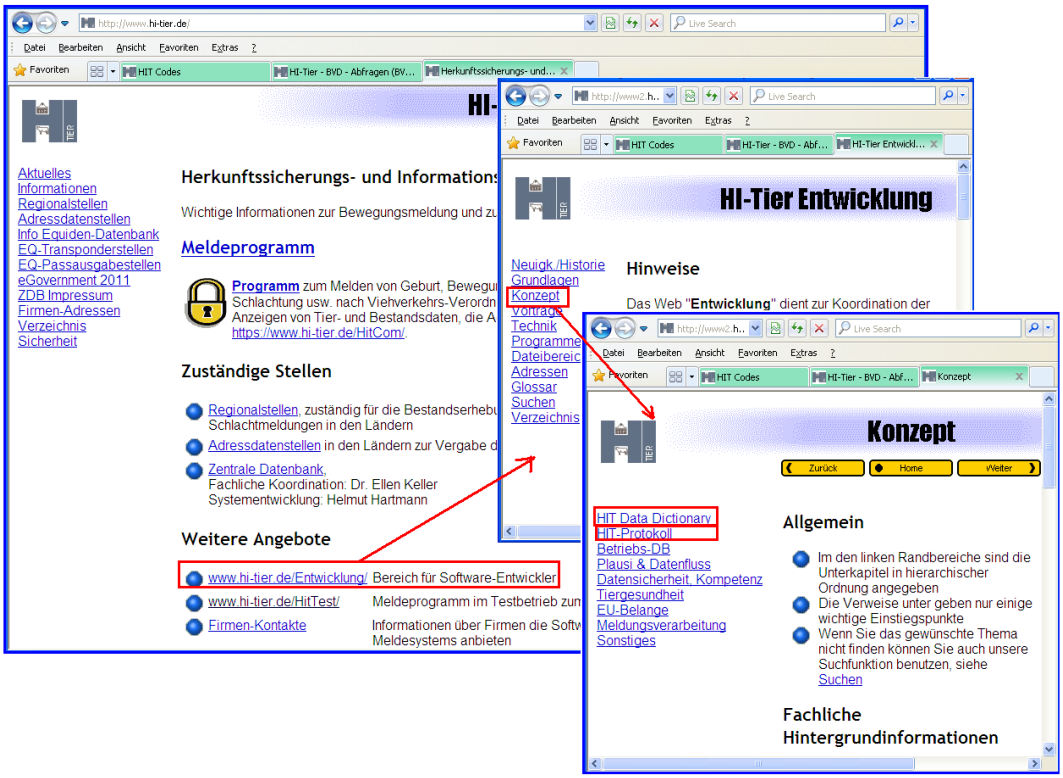
*<Nr>:<Aktions-Code><Stückelung>/<Sub-Code>:<Meldung>/<Feldliste>:<Datenliste>

http://www.hi-tier.de/Entwicklung/Konzept/HITP/feink040.html

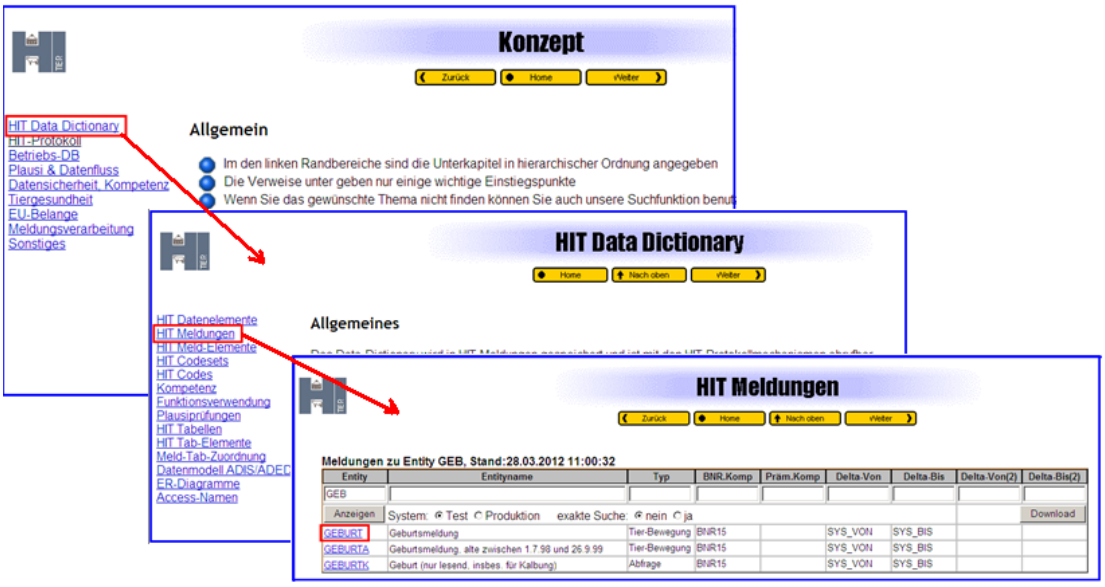
Syntax und Bestandteile

Token	Definition	Bemerkung
<Befehl>	<**Einleitung**>:<**Aktion**>:<**Objekt**>:<**Datenliste**>	4 Haupt-Elemente, mit Doppelpunkt getrennt
<**Einleitung**>	<Befehls-Kenner><Befehls-Nummer>	
<Befehls-Kenner>	* (Stern) | + (Pluszeichen)	kennzeichnet Befehlsbeginn, + wenn weitere Befehlsteile folgen, * beim letzten oder einzigen Befehlsteil
<Befehls-Nummer>	<Haupt-Nummer>[+<Unter-Nummer>][#<Rowkeys>]	Achte auf Trennung der Unter-Nummer durch '+'
<Haupt-Nummer>	Fortlaufende Nummer des Befehl	dient zur richtigen Zuordnung des Ergebnisses und Sicherung der Reihenfolge und Vollständigkeit, beginnt bei 1 für die Connection
<Unter-Nummer>	Fortlaufende Nummer des Teilbefehls, wenn der Befehl aus mehreren Zeilen besteht	dient zur Sicherung der Reihenfolge und Vollständigkeit wenn Teilbefehle vorliegen, beginnt bei 1 für jede neue Haupt-Nummer
<Rowkeys>	<Rowkey>[;<Rowkeys>]	Liste von benutzerdefinierten Keys
<Rowkey>	benutzerdefinierten Key (String-Konstante)	dient zur Identifikation des Befehls
<**Aktion**>	<Aktions-Code><Stückelung>[/<Sub-Codeliste>]	
<Aktions-Code>	X | I | U | S | D | R	X (=Excecute) | I (=Insert) | U (=Update) | S (=Strono) | D (=Delete) | R (=Retrieve) | C (=Confirm)
<Stückelung>	F | S | B | T	F (=Feld) | S (=Satz) | B (=Block) | T (=Transaktion)
<Sub-Codeliste>	<Sub-Code>[;<Sub-Codeliste>]	ein Sub-Code oder mit Semikolon getrennt mehrere Sub-Codes
<Sub-Code>	T | E | S | A | P | O | ...	T (=Trotzdem) | E (=ENDE) | S (=SICHER) | A (=Abbruch) | P (=Panik) | O (=NOOP) ... siehe Sub-Codes, für RETRIEVE gibt es noch weitere Sub-Codes, siehe HIT-QL
<**Objekt**>	[<Meldung>][/<Feldliste>]	Meldung kann bei nachfolgenden Feldern der selben Meldung fehlen
<Meldung>	LOGON | BETRD | GEBURT |	Feste Strings, siehe Meldung
<Feldliste>	<Feldname>[;<Feldliste>]	ein Feld oder mit Semikolon getrennt mehrere Felder
<Feldname>	BNR15 | PIN | ...	komplette Liste je nach Meldung, siehe Meldungselemente
<**Datenliste**>	<Datenelement>[;<Datenliste>]	ein Datenwert oder mit Semikolon getrennt mehrere Datenwerte

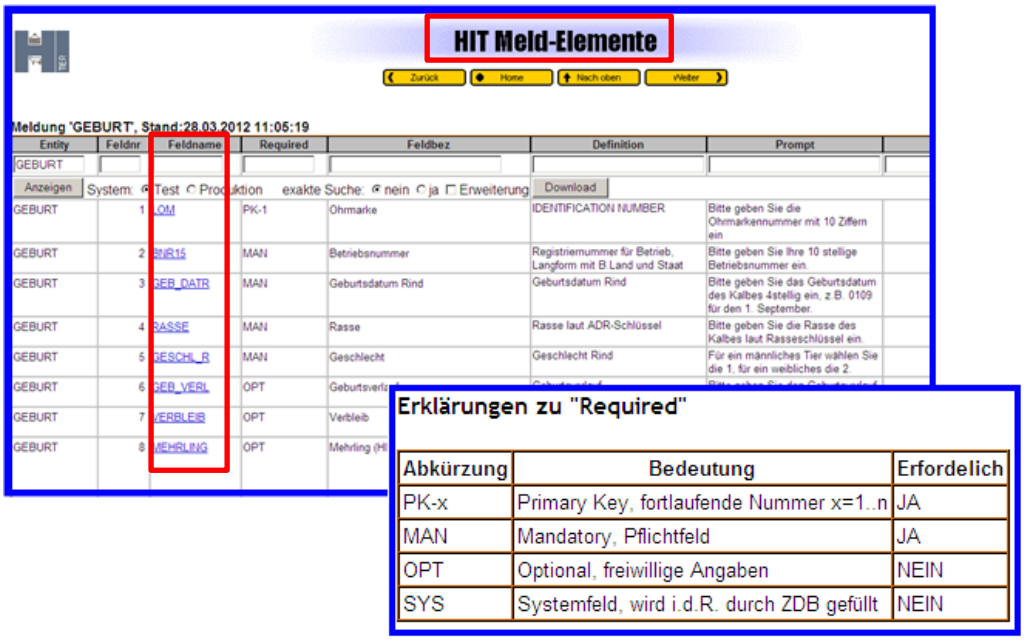
Im „**HIT Data Dictionary**“ sind die Melde-Entitäten, Codesets, Plausibilitätsprüfungen usw. zu finden. In „Meld-Elemente“ werden die „Feldnamen“ für die einzelnen Meldungen, die für die Befehle erforderlich sind, aufgelistet.



Oder direkt über den Link: http://www.hi-tier.de/Entwicklung/Konzept/Default.htm



Klick auf Meldung (z.B. Geburt) führt zu den Meld-Elementen (Feldnamenliste)



„Required“ Felder der Meldungen beachten (Pflichtfelder/optionale Felder)!

2.2.1.2. Beispiele von Meldungen im UnitTest

Für ein Tier Geburt melden (*2), dann Todmeldung (*3), dann Datum der Todmeldung ändern (*4)

4 Befehlsteile jeweils mit Doppelpunkt getrennt

„Einleitung“:	**„Aktion“:**	**„Objekt“: Meldung und zugehörige Felder**	**„Datenliste“:**
*2:	IS/S: (Insert)	GEBURT/GEB_DATR;RASSE;GESCHL_R;LOM;LOM_MUT;BNR15:	01.01.2012;1;2;276000913105113;276000913105100;276091310500001
*3:	IS:	TOD/LOM;TOD_DAT;TOD_ART;BNR15:	DE 010000163;02.01.2012;1;276000913105113
*4:	XS: (Execute)	TOD/LOM;TOD_DAT;TOD_ART;BNR15:	DE 010000163;05.01.2012;1;276000913105113

Befehl in **Testcase.txt** schreiben :

- „;“ = Kommentar, bei den Befehlen, die im UnitTest laufen sollen vorne entfernen
- Name der Entität (Meldung) suchen
  http://www.hi-tier.de/Entwicklung/Konzept/_asp/dd00001.asp
- Feldnamen aus Meld-Elemente
  http://www.hi-tier.de/Entwicklung/Konzept/_asp/dd00002.asp

```
; Demo: Test einer Geburtsmeldung

; Anmelden
*1:XS:LOGON/BNR15;PIN;MELD_WG;TIMEOUT:276091310500015;900015;3;1200

; Geburtsmeldung liefert Schwere 3, da Daten bereits vorhanden
;*2:IS/S:GEBURT/GEB_DATR;RASSE;GESCHL_R;LOM;LOM_MUT;BNR15:%HeuteMin3MonateRoundDown%;1;2;276000913105103;276000913105100;276091310500001
*2:IS/S:GEBURT/GEB_DATR;RASSE;GESCHL_R;LOM;LOM_MUT;BNR15:01.01.2012;1;2;276000913105113;276000913105100;276091310500001
*3:IS:TOD/LOM;TOD_DAT;TOD_ART;BNR15:DE 01 000 00163;02.01.2012;1;276000913105113
*4:XS:TOD/LOM;TOD_DAT;TOD_ART;BNR15:DE 01 000 00163;05.01.2012;1;276000913105113
; Festlegen, daß Schwere 3 erwartet wird:
#SCHWERE_EQ 3

; Abmelden
*99:XS:LOGOFF/:


#ECHO 1
```

- „runJUnit.cmd“ laufen lassen

=> grüner Balken = OK, Test ist durchgelaufen / roter Balken „testcase.log“ ansehen „Test.log“ gibt aus, was getan wurde

LOG-Datei

```
Datei Bearbeiten Suchen Ansicht Kodierung Sprachen Einstellungen Makro Ausführen Erweiterungen Fenster ?
change.log | testcase.ini | testcase.txt | testcase.log
12  MinVorgNr=900000000;
13  MaxVorgNr=999999999;
14  StartVorgNr=900000000;
15  ConfirmStartTimestamp=2001-01-01 00:00:00.0;
16  VetMeldStartDate=2003-01-01;
17  [Webpage]
18  ProxyHost=www-proxy.bybn.de;
19  ProxyPort=80;
20  [Database]
21  UseDbAccess=1;
22  JdbcClass=COM.ibm.db2.jdbc.app.DB2Driver;
23  JdbcService=jdbc:db2:;
24  SqlDsn=HITDB2;
25  SqlUserid=AXDB202;
26  SqlPassword=********;
27  ----------
28
29  Verarbeitet testcase.txt
30  ------------------------------------------------
31              |#ECHO wurde auf 2 gesetzt (protokolliert Befehlszeile mit HIT-Status).
32  Zeile    24|*1:XS:LOGON/BNR15;PIN;MELD_WG;TIMEOUT:276091310500015;900015;3;1200
33              |Status: *1:XS:LOGON/BNR15;PIN;MELD_WG;TIMEOUT:276091310500015;900015;3;1200
34              |Plausi: 231;223
35              |Error : %1%1:1/231:LOGON/BNR15;MELD_WG:Eine RS sollte den Meldeweg POST verwenden.
36              |        =1%2:0/223:LOGON/*:Anmeldung erfolgreich.
37  Zeile    28|*2:IS/S:GEBURT/GEB_DATR;RASSE;GESCHL_R;LOM;LOM_MUT;BNR15:01.01.2012;1;2;276000913105113;276000913105100;276091310500001
38              |Status: *2:IS/S:GEBURT/GEB_DATR;RASSE;GESCHL_R;LOM;LOM_MUT;BNR15:01.01.2012;1;2;276000913105113;276000913105100;276091310
39              |Plausi: 308;417;130
40              |Error : %2%1 1/308:GEBURT/GEB_DATR:Die Meldefrist von 7 Tagen ist überschritten, bitte Frist künftig beachten.
41              |        %2%2 1/417:GEBURT/LOM_MUT:Kalbung liegt weniger als 210 Tage zurück.Ohrmarke Muttertier prüfen oder Geburt neu me
42              |        =2%3 0/130:GEBURT:Die Meldung wurde abgespeichert.
43  Zeile    29|*3:IS:TOD/LOM;TOD_DAT;TOD_ART;BNR15:DE 01 000 00163;02.01.2012;1;276000913105113
44              |Status: *3:IS:TOD/LOM;TOD_DAT;TOD_ART;BNR15:DE 01 000 00163;02.01.2012;1;276000913105113
45              |Plausi: 2057;1011;284
46              |Error : %3%1:1/2057:TOD/TOD_DAT:Die Meldefrist von 7 Tagen ist überschritten, bitte Frist künftig beachten.
47              |        %3%2:3/1011:TOD:Sie haben keine Berechtigung da diese Daten nicht in ihrem Bereich liegen
48              |        =3%3:2/284:TOD/LOM:Für diese Ohrmarkennummer ist bereits eine Todesmeldung von einem anderen Betrieb vorhanden.
49  Zeile    30|*4:XS:TOD/LOM;TOD_DAT;TOD_ART;BNR15:DE 01 000 00163;05.01.2012;1;276000913105113
50              |Status: *4:XS:TOD/LOM;TOD_DAT;TOD_ART;BNR15:DE 01 000 00163;05.01.2012;1;276000913105113
51              |Plausi: 1011;284
52              |Error : %4%1 3/1011:TOD:Sie haben keine Berechtigung da diese Daten nicht in ihrem Bereich liegen
53              |        =4%2 2/284:TOD/LOM:Für diese Ohrmarkennummer ist bereits eine Todesmeldung von einem anderen Betrieb vorhanden.
```

Error= Ergebnis

3/ 1011= Fehlerschwere mit Plausinummer und Beschreibung

Plausi-Schwere >1 verhindert Speicherung der Daten!

2.2.2. HIT-Abfragen in HIT-QL (Query-Language)

http://www.hi-tier.de/Entwicklung/Konzept/HITP/feink041.html

Die HIT-QL (**Query-Language**) ist die formale Beschreibung einer einfachen Datenbankabfragesprache. Es geht dabei um das Holen von Daten aus tabellenartigen Strukturen durch Auswahl der gewünschten Tabelle oder Meldungsart (Entität), Auswahl oder Einschränkung der gewünschten Spalten (Meld-Elemente/Feldliste) und Einschränken der Datenzeilen durch Suchbedingungen sowie ggf. Sortieren und Gruppieren des Ergebnisses und Ermitteln abgeleiteter Werte wie Zeilenzahl oder Min/Max-Werte. Der Zugriff auf die verschiedenen Entitäten sowie Daten innerhalb der Tabellen wird durch die Kompetenz des Abfragenden beschränkt.

Befehlsaufbau in Kürze:

<Einleitung>	**<Aktion>:**	**<Objekt>:**	**<Queryelemente>:**
Analog HIT-Spezifikation	<Aktions-Code> **R** (=Retrieve); <Stückelung> **S** (=Satz)	Analog HIT-Spezifikation	<Bedingungselement>[<Sortierung>;<Gruppierung>>;<Kommentar>]Bestandteile der Abfrage
*3:	RS:	TIERSTATX/LOM; GEB_DAT:	ST_LEBEND;=;1;AND;BNR15_LMON;=;09136000091

Beispiel mit Operatoren:

Lebende Tiere mit BVD-Status P10 und P11 in NRW:

*1: RS: TIERSTATX/LOM:BNR15_LMON **IS** 05 AND BVD_ERG **IN** (410, 411) **AND** ST_LEBEND;=;1

Operator-Liste

Operator	Vergleichsziel	oder	Definition	Beschreibung	entspricht
EQ	<Feldinhalt>|<Konstante>	=	EQUAL	gleich	=
EQX	<Feldinhalt>|<Konstante>		EQUAL EXTENDED	gleich, case insensitiv	
NE	<Feldinhalt>|<Konstante>	^=	NOT EQUAL	nicht gleich	^=
NEX	<Feldinhalt>|<Konstante>		NOT EQX	nicht gleich, case insensitiv	
LT	<Feldinhalt>|<Konstante>	<	LESS TO	kleiner als	<
LE	<Feldinhalt>|<Konstante>	<=	LESS EQUAL	kleiner gleich	<=
GT	<Feldinhalt>|<Konstante>	>	GREATER TO	größer als	>
GE	<Feldinhalt>|<Konstante>	>=	GREAT EQUAL	größer gleich	>=
IN	<Konstantenliste>	IN	IN	in nachfolgender Liste	IN (x , y , z)
NI	<Konstantenliste>	NOT IN	NOT IN	nicht in Liste	NOT IN (x , y , z)
BW	<Konstante>;<Konstante>	BETWEEN	BETWEEN	zwischen x und y	BETWEEN x AND y
NB	<Konstante>;<Konstante>	NOT BETWEEN	NOT BETWEEN	nicht zwischen	NOT BETWEEN
LI	<Konstante> mit Wildcard "%25"	LIKE	LIKE	gleich mit Wildcard/Joker	
LIX	<Konstante> mit Wildcard "%25"	LIKEX	LIKE EXTENDED	gleich mit Wildcard/Joker, case insensitiv	
NL	<Konstante> mit Wildcard "%25"	NOT LIKE	NOT LIKE	nicht gleich mit Wildcard/Joker	
NLX	<Konstante> mit Wildcard "%25"	NOT LIKEX	NOT LIKE EXTENDED	nicht gleich mit Wildcard/Joker, case insensitiv	
IS	<erweiterte Vergleichsbedingung>	*	Kontext spezifisch	in der Vergleichsbedingung können Werte, Listen und erweiterte Bereiche angegeben werden, darunter auch Landkreisbereichskürzel und KFZ-Kennzeichen. Genaue Beschreibung siehe bei "Komplexe Vergleichsbedingung" (Status: experimentell).	

Verknüpfungs-Liste

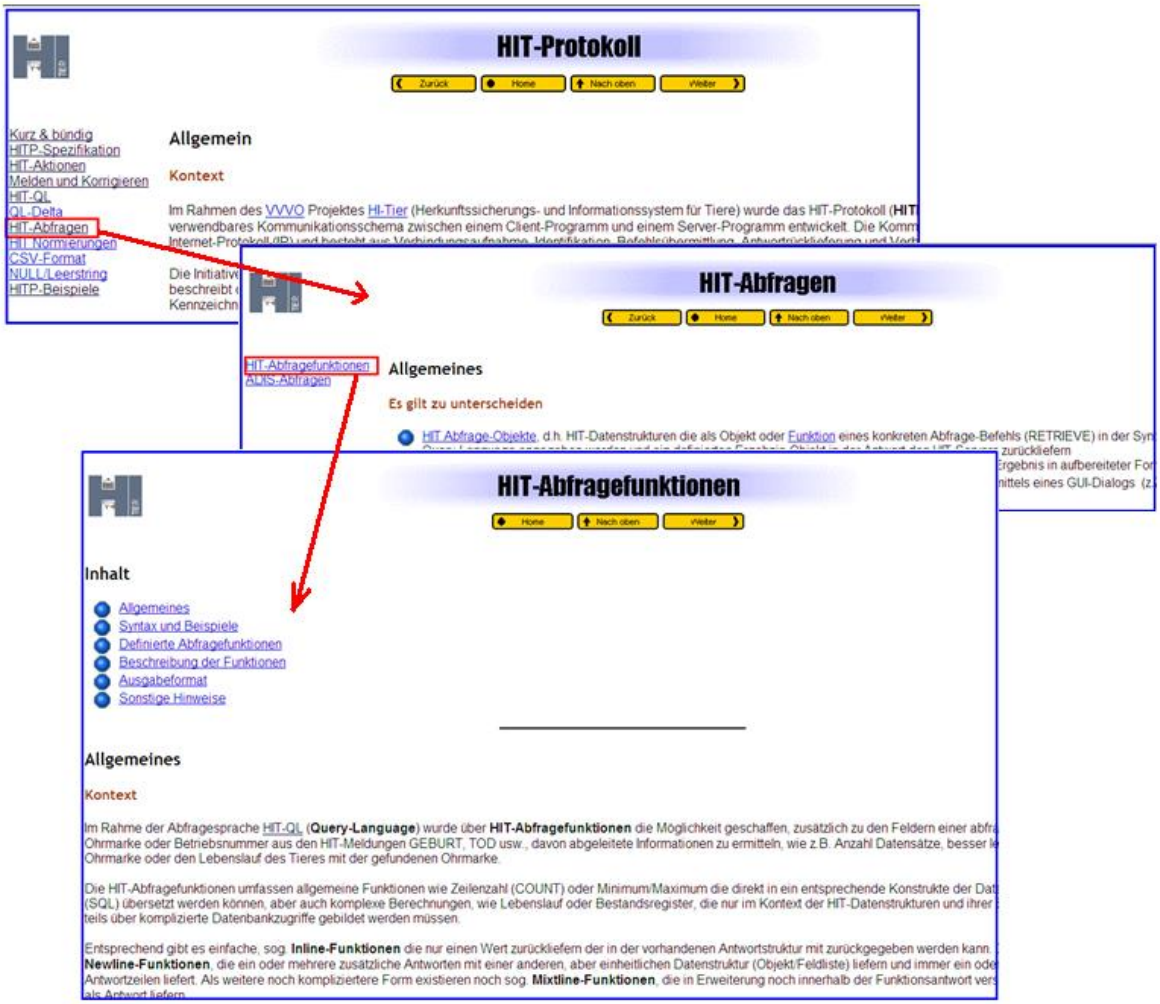
Operator	Definition	Beschreibung
AND	logisches UND	und zugleich
OR	logisches ODER	oder
NAND	logisches UND NICHT	und nicht
NOR	logisches ODER NICHT	oder nicht

Spezielle RS- Abfragen

RS/**I**	**I**nclusive stornierte Daten Beispiel: : *2:RS/I:BTR_D/*:BNR15;=;090000000001 Normalerweise werden bei Tabellen, die Historisierung unterstützen, nur aktuelle Sätze angezeigt (BIS-Timestamp offen, bzw. 31.12.2100). Mit dem Subcode I erhält man alle Sätze, inklusive der Stornierten.
RS/**Z**	Abfrage zu einem bestimmten **Z**eitpunkt
RS/**D**	**D**elta-Abfrage von Daten, die nach der letzten Abfrage hinzugekommen sind
RS/**H**	Delta-Abfrage und **H**istorie (auch stornierte Meldungen)

2.2.3. HIT-Abfragefunktionen

http://www.hi-tier.de/Entwicklung/Konzept/HITP/feink045.html



Im Rahmen der Abfragesprache HIT-QL (Query-Language) wurde über HIT-Abfragefunktionen die Möglichkeit geschaffen, zusätzlich zu den Feldern einer abfragbaren Entität davon abgeleitete Informationen, wie z.B. Anzahl Datensätze oder besser lesbare Alphadarstellungen der Ohrmarke/Betriebsnummer zu ermitteln (276000091234567 wird zu DE 09 123 4567).

Inline-Funktionen (mit $ gekennzeichnet)

→liefern nur einen Wert zurück

Beispiel:

Zählen der Betriebe und Tiere mit Status U0 in einem Land, z.B. Niedersachsen

```
*1:RS:$COUNTD(BNR15_LMON/Anz_Betriebe);$COUNT(*/Anz_Tiere):
BNR15_LMON;BW;276030000000000;276039999999999;AND;BVD_ERG;EQ;300
```

Count => zählt die Zeilen

CountD => zählt verschiedene Zeilen (Betrieb wird dadurch nur einmal gezählt)

Newline-Funktionen (mit # gekennzeichnet, z.B. #bestreg)

→ die ein oder mehrere zusätzliche Antworten mit einer anderen, aber einheitlichen Datenstruktur (Objekt/Feldliste) liefern und immer ein oder mehrere neue Antwortzeilen liefern.

Mixtline-Funktionen (mit ! gekennzeichnet)

→ die in Erweiterung noch innerhalb der Funktionsantwort unterschiedliche Datenstrukturen als Antwort liefern.

Funktionstabelle, siehe http://www.hi-tier.de/Entwicklung/Konzept/HITP/feink045.html

Funktion	Beschreibung
$COUNT(*)	Zeilenanzahl
$COUNTB(*)	Zeilenanzahl, bei sehr großen Datenmengen
$COUNTF(RASSE)	Zeilenanzahl mit vorhandener Rasse
$COUNTD(BNR15)	Anzahl verschiedener Betriebe
$COUNT(*/ANZ_GEBURTEN)	Zeilenanzahl, als Ergebnis-Spalte "ANZ_GEBURTEN"
$MIN(GEB_DATR)	kleinstes Geburtsdatum, Outputname ist Standard
$MIN(GEB_DATR/FIRST_DAT)	kleinstes Geburtsdatum, explizite Angabe des Outputnamens
$YEAR(GEB_DATR/JAHR_GEB)	Jahr aus dem Geburtsdatum, explizite Angabe des Outputnamens
$MONTH(GEB_DATR)	Monat aus dem Geburtsdatum
$DAY(GEB_DATR)	Tag aus dem Geburtsdatum
$DAYS(GEB_DATR)	Laufende Nummer des Tag seit 1.1.1970
$DATE(SYS_VON)	Datum aus dem Timestamp SYS_VON
$ALPHALOM(LOM_MUT)	Ohrmarke in aufbereiteter Form
$LOMSTAAT(LOM)	Staat (numerisch) aus der Ohrmarke, z.B: 276 für DE
$LOMLAND(LOM)	Land (numerisch) aus der Ohrmarke, z.B: 16 für Thüringen
$BNRSTAAT(BNR15)	Staat (numerisch), z.B: 276 für DE bei Betrieb 09 177 148 0001
$BNRLAND(BNR15)	Land (numerisch), z.B: 9 bei Betrieb 09 177 148 0001
$BNRREGBEZ(BNR15)	Regierungsbezirk (numerisch), z.B: 1 bei Betrieb 09 177 148 0001
$BNRKREIS(BNR15)	Landkreis (numerisch), z.B: 177 bei Betrieb 09 177 148 0001
$BNRGEMDE(BNR15)	Gemeinde (numerisch), z.B: 148 bei Betrieb 09 177 148 0001
$MELD_FRST(ZUGA_DAT)	Berechnet Differenz aus Meldedatum und Ereignisdatum
$MELD_FR14(ZUGA_DAT)	Differenz aus Meldedatum und Ereignisdatum, beschränkt auf 14 Tage
$MELD_FRG1(ZUGA_DAT)	Differenz aus Meldedatum und Ereignisdatum, nach Gruppen 1
$MELD_FRG2(ZUGA_DAT)	Differenz aus Meldedatum und Ereignisdatum, nach Gruppen 2
$DATE_DIFF(VOK_ADAT;VOK_ADAT)	Differenz aus VOK-Datum und Datum der Ankündigung
$STORNO(SYS_BIS)	Ermittelt ob Datensatz storniert ist, also Wert des Timestamps 31.12.2100
$AKTIV(SYS_BIS)	Ermittelt ob Datensatz aktiv ist ist, also Wert des Timestamps größer jetzt (current timestamp)
#TIEREIN(BNR15;LOM;ZUGA_DAT)	Tiereingang, default alle Ergebnisspalten
#TIEREIN(*/*)	Tiereingang, explizit alle Ergebnisspalten
#BESTREG(27.9.1999;31.12.1999/*)	Bestandsregister zum Stichtag, alle Ergebnisspalten
#BESTREG(*/ANF_TYP;ANF_DAT)	Default-Argument Bestand heute, nur bestimmte Ergebnisspalten

Komplexe Funktionen (Newline/ Mixtline) dürfen nur mit echten Tabellen verwendet werden. Eine Führungsentität muss gewählt werden.

Funktionen können im RETRIEVE nicht als Objekt-Meldung sondern nur als Objekt-Feldelement benutzt werden. Welche Meldungsart bei der Abfrage damit zusammen mit einer bestimmten Funktion verwendet werden kann siehe unter Funktionsverwendung/Führungs-Entitäten.
http://www.hi-tier.de/Entwicklung/Konzept/_asp/dd00011.asp



HIT Data Dictionary

Home | Nach oben | Weiter

HIT Datenelemente
HIT Meldungen
HIT Meld-Elemente
HIT Codesets
HIT Codes
Kompetenz
Funktionsverwendung
Plausiprüfungen
HIT Tabellen
HIT Tab-Elemente
Meld-Tab-Zuordnung
Datenmodell ADIS/ADED
ER-Diagramme
Access-Namen

Allgemeines

Das Data-Dictionary wird in HIT-Meldungen gespeichert und ist mit den HIT-Protokollmechanis

Standard Darstellung

- Datenelemente - alle Datenfelder
- **Meldungen** - HIT-Entitäten für Datenlieferung, Abfrage und Metadatenabfrage (**der best**
- Meldungs-Elemente - Meldungen mit allen den Feldern
- Codesets - Übersicht über die Schlüssellisten
- Codes - zur Darstellung der **Schlüsselwerte**
- Kompetenz - zur Anzeige der **Kompetenzen** für die Meldungen, Aktionen und Betriebsty
- Plausiprüfungen - alle Plausibilitätsprüfungen



Funktionsverwendung

Zurück | Home | Nach oben | Weiter

Funktionsverwendungen zu FUNKTION #BEST, Stand:28.03.2012 15:41:03

FUNKTION	ENTITY *1	PARMLIST *2
#BEST		
Anzeigen	System: Test Produktion	exakte Suche: nein ja Download
#BESTREG	BCFORBPX	<date>;<date>;BNR15
#BESTREG	BCFORBPX	<date>;<date>;BNR15_P
#BESTREG	BETRD	<date>;<date>;BNR15
#BESTREG	BETRD_R	<date>;<date>;BNR15
#BESTREG	BETRD_T	<date>;<date>;BNR15
#BESTREG	BETRR	<date>;<date>;BNR15
#BESTREG	BETRT	<date>;<date>;BNR15
#BESTREG	BETRZ	<date>;<date>;BNR15_C
#BESTREG	BPFORBCX	<date>;<date>;BNR15
#BESTREG	BPFORBCX	<date>;<date>;BNR15_C
#BESTREG	BSFORUN	<date>;<date>;BNR15
#BESTREG	BTR_D	<date>;<date>;BNR15
#BESTREG	BTR_D_AKT	<date>;<date>;BNR15
#BESTREG	BTR_D_OLD	<date>;<date>;BNR15
#BESTREG	BTR_D_R	<date>;<date>;BNR15
#BESTREG	BTR_D_T	<date>;<date>;BNR15
#BESTREG	BTR_R	<date>;<date>;BNR15
#BESTREG	BTR_R_AKT	<date>;<date>;BNR15
#BESTREG	BTR_R_OLD	<date>;<date>;BNR15
#BESTREG	BTR_T	<date>;<date>;BNR15
#BESTREG	BTR_T_AKT	<date>;<date>;BNR15
#BESTREG	BTR_T_OLD	<date>;<date>;BNR15

2.2.4. HITP-Beispiele

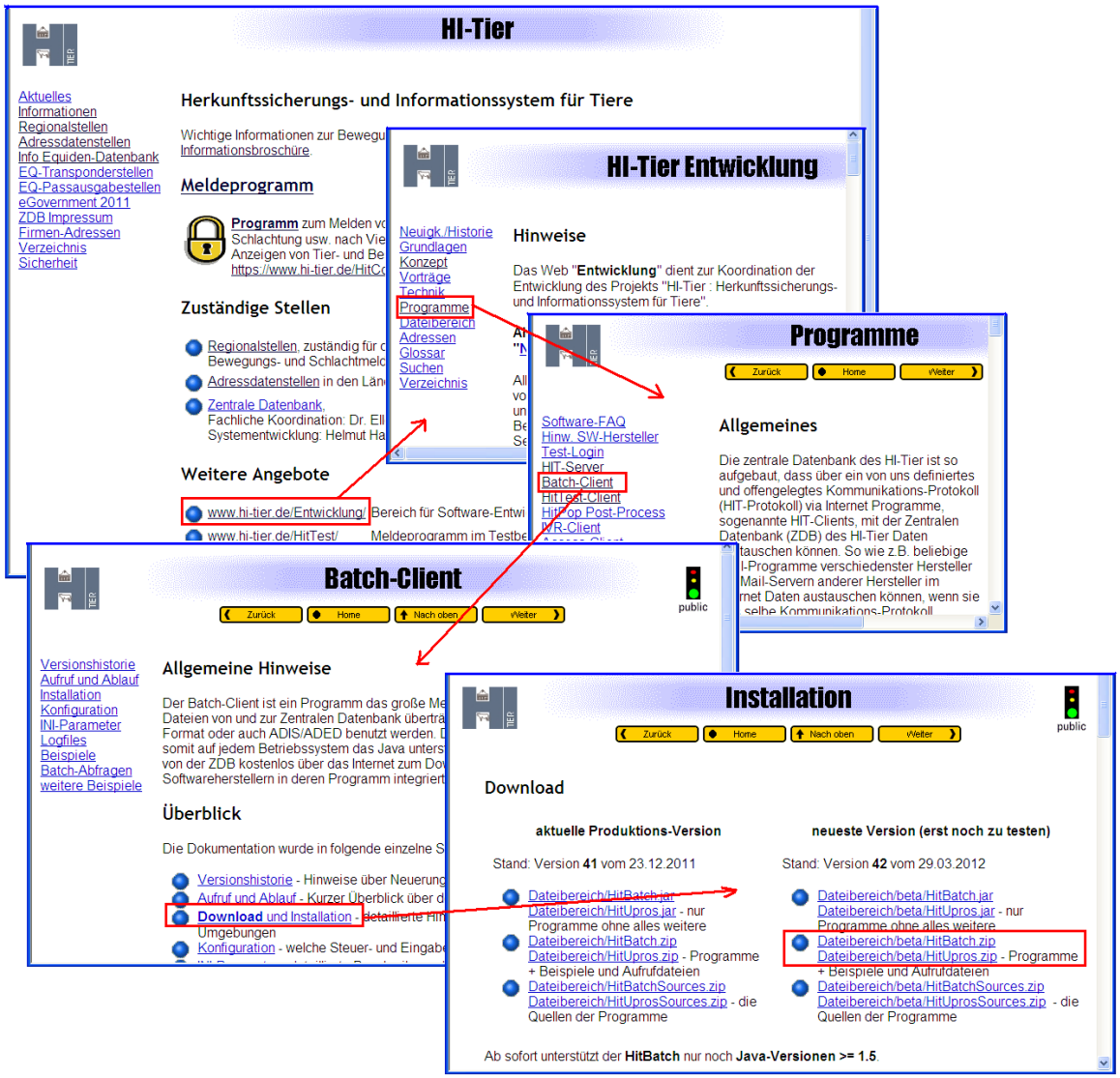
http://www.hi-tier.de/Entwicklung/Konzept/HITP/bsp.html

Zusammenstellung einiger Beispiele von Meldungen und Abfragen in HIT-Protokoll



3. HIT-Batch

3.1. Technische Voraussetzungen



Oder Link direkt: http://www.hi-tier.de/Entwicklung/Programme/batch005.html

- Download „**HitBatch.zip**“ und „**Upros.zip**“, beide öffnen, ZIP-Inhalt in gemeinsamen Ordner
- ZIP-Datei extrahieren
- HITBATCH**.cmd**: ausführende Datei („command“)→ Start der Abfrage
  - in Texteditor öffnen, unter „ set BASEDIR“ Pfad dieses Orders angeben, z.B. C:\HIT
- HITBATCH**.ini:** Initialisierungsdatei - Steuerung und Konfigurationseinstellungen (INI-Parameter, siehe http://www.hi-tier.de/Entwicklung/Programme/batch002.html)
  - in Texteditor öffnen
  - Mit !!! Parameter prüfen bzw. anpassen
  - User-ID (Betriebsnummer und PIN), MBN, Mandant anpassen
  - PIN als txt-Datei anlegen, Pfad zur Datei bei PIN angeben
    PIN-Automatik: Für automatisierte Vorgänge => wenn PIN abläuft wird vom System automatisch neue PIN vergeben und in Datei geschrieben. Nur geeignet für „reine“ HIT-Batch-Zugänge, da diese PIN nicht einsehbar.

- Port wie bei **Testcase.ini** (siehe 2.1.2.1)
- Schwere 0 => alles, auch Erfolg wird mitgeschrieben
- Man kann bis zu 50 verschiedene Arbeitsaufträge= SET bearbeiten. Wenn „setcout“ = 1 führt er nur den 1. Set aus

➢ Datei.**in**: beinhaltet die Abfrage
- Zeile 1: <AUSGABEFELDER>
- Zeile 2 : <BEDINGUNG>

➢ Datei.**log**: Transaktionsprotokoll (LOG-Datei prüfen, insbesondere beim Senden von Daten→Fehlermeldungen)

HITBatch.CMD

```
1  @echo off
2  REM
3  REM Aufrufscript
4  REM
5
6  cd /d %~dp0
7
8  set BASEDIR=\projekt\HIT
9
10
11 :AGAIN
12 set HBINI=%1
13 if %HBINI%.==. set HBINI=HitBatch.ini
14 if exist %HBINI% goto RUNHB
15 echo Parameter-Datei: %HBINI% nicht gefunden
16 goto FERTIG
```

Pfad des Ordners angeben, in dem die herunter geladenen HitBatch.zip“ und „Upros.zip gespeichert sind!

HITBatch.INI



User-ID= In Testdatenbank die Test-Betriebsnummer mit entsprechender Kompetenz wählen (hier als Regionalstelle- 15); in Produktion die eigene Betriebsnummer eintragen.

Für **PIN** ggf. Ordner anlegen; verschlüsselte PIN-Übertragung mit # →PIN=#I:\HIT-Batch-Schulung\PIN

Server prüfen:
Adressen außerhalb des bayer. Behördennetz:
PRIMARYSERVER=hitserver.hi-tier.de
PRIMARPORT=2223 (Test) oder 2222 (Produktion)
BACKUPSERVER=hitbackup.hi-tier.de

Schwere=0 (auch Erfolgsmeldung wird ausgegeben)
Bei der „Schwere“ wird festgelegt, welche Hinweise/Nachfragen/Fehler in Log-Datei ausgegeben werden

Setcount= Anzahl der auszuführenden Aufträge (Set)
Startset= Nummer des Set, bei dem der HitBatch beginnen soll

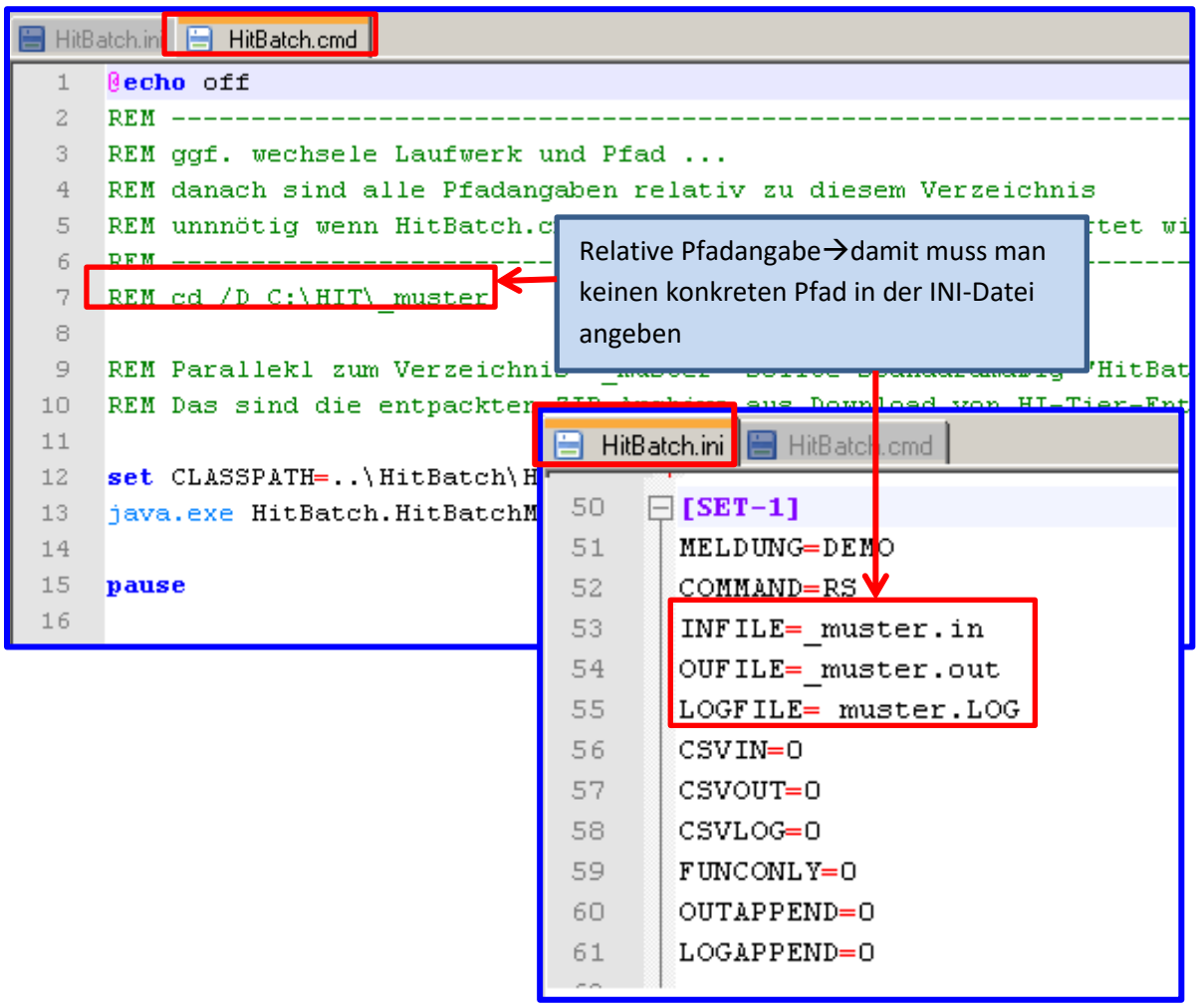
3.2. Alternativ: Vereinfachte Version HIT-Batch

http://www.hi-tier.de/entwicklung/Programme/batch008.html

(INI-Datei auf wesentliche Struktur reduziert -ohne Kommentare und Beispielabfragen)



Muster



3.3. HIT-Batch-Abfragen

http://www.hi-tier.de/Entwicklung/Programme/batch001.html

Eine HIT-Abfrage wird über den R(etrieve)-Befehl ausgeführt. Man gibt die Tabelle an, die abgefragt werden soll, dann die Felder, die ausgegeben werden sollen und schließlich noch die Abfragebedingung. Diese vier Teile gibt man wie folgt an (vergleiche Hit-Protokoll):
*x:<BEFEHL>:<TABELLE>/<AUSGABEFELDER>:<BEDINGUNG>

Die Ausgabefelder können zudem Abfragefunktionen wie $ALPHALOM() oder #BESTREG() enthalten.

Der HitBatch ist eigentlich für Abfragen nicht gedacht, da er nur das Melden von Daten vereinfachen soll. Es lassen sich aber dennoch Abfragen damit ausführen, jedoch mit den Formaten, die zum Melden gedacht sind.
Die Ini-Datei muss dazu in einem Set analog zu den Meldungen den Parameter MELDUNG passend setzen, als COMMAND entweder einen Stern '*' (nur wenn der COMMAND in der Datendatei INFILE dabei ist) oder ein 'RS' (evtl. incl. Subcodes) und OUFILE kann auf eine Ausgabedatei verweisen, in das das Ergebnis der Abfrage geschrieben wird.

3.3.1. HITP-Vorgehen: Senden mehrerer gleicher Meldungen

Formulierung im HIT-Protokoll (HITP)

*2:IS:TOD/LOM;BNR15;TOD_DAT:DE0100000001; 01 000 000 0001;01.02.2012
*3:IS::DE 01 000 00002; 090000000001; 1.2.2012
*4:IS::276000100000003; 276090000000001; 1.2.12

Im Prinzip macht Batch nichts anderes als eine Wiederholung der Meldung/Abfrage wie HITP.

HitBatch.INI zum Batch-Senden (IS):

[SET-1]
Command=IS
Meldung=TOD
Infile=TOD.csv (CSV-Datei mit Endung vorzugsweise .csv oder .in oder .txt)

TOD.csv enthält die zu übermittelnden Daten
Zeile 1 | LOM;BNR15;TOD_DAT
Zeile 2 | DE0100000001; 01 000 000 0001;01.02.2012
Zeile 3 | DE 01 000 00002; 09000000 0001; 1.2.2012
Zeile 4 | 276000100000003; 276090000000001; 1.2.12

3.3.2. HITP-Vorgehen: Abfragen mit Query-Language (HIT-QL)

Beispiel:
Anzahl Tiere mit BVD-Status P9 und P99 in Land NDS , gruppiert nach Status (lebend /tot/sonstiges)
Formulierung im HIT-Protokoll (HITP)

*2:RS:TIERSTATX/ST_LEBEND;$COUNT(*):BLAND_LMON;=;03;AND;BVD_ERG;BW;409;499;
GROUP;ST_LEBEND

HitBatch.INI für Abfrage (RS):

[SET-1]
Command=RS
MELDUNG=TIERSTATX
INFILE=I:\HIT-Batch-Schulung \Abfrage1.in (CSV-Datei mit Endung vorzugsweise .in oder .txt oder .csv)
OUFILE=I:\HIT-Batch-Schulung \Abfrage1.OUT
LOGFILE=I:\HIT-Batch-Schulung \Abfrage1.LOG
OUTAPPEND=0 (legt fest, ob beim Aufruf des SET die Daten im OUFILE überschrieben oder hinten angehängt)
LOGAPPEND=0 (legt fest, ob beim Aufruf des SET die Daten im LOGFILE überschrieben oder hinten angehängt)
CSVIN=0 (Format für Eingabedatei festlegen)
CSVOUT=0 (Format für Ausgabedatei festlegen)

Abfrage1.in (in Zeile 1 sind die Ausgabefelder, in Zeile 2 ist die Bedingung)
Zeile 1 | ST_LEBEND;$COUNT(*)
Zeile 2 | BLAND_LMON;=;09;AND;BVD_ERG;BW;409;499; GROUP;ST_LEBEND

3.3.3. Filterfunktionen

http://www.hi-tier.de/entwicklung/programme/batch002.html

spezielle Aktion zum Filtern von Dateien	
`FilterArt`	Spezielle Angabe ob es sich bei diesem Step um einen normalen HIT-Batch Verarbeitungsschritt handelt bei dem HIT-Protokollbefehle aus dem IN-File an den Server zum Senden, Ändern oder Lesen von Daten geschickt werden oder ob nur lokal auf dem IN-File Kopier- und Filterverarbeitung durchgeführt werden soll.: **0**: NORM - Normaler HIT-Batch Verarbeitungsschritt (Default) **1**: COPY - reines Kopieren der Datensätze aus der Inputdatei (INFILE) in die Outputdatei (OUFILE) ggf. anwenden von post-output-processing aus optionaler POP-Steuerdatei (POPFILE) **2**: FILTER - Klassicher Filter, kopiere nur Daten aus der Inputdatei (INFILE) in die Outputdatei (OUFILE) wenn die Daten auch in der Filterdatei (FILTERFILE) vorkommen, also "copy IN to OUT if exist in FILTER" **3**: INVERS - Klassicher Filter (andersrum), kopiere nur Daten aus der Filterdatei (FILTERFILE) in die Outputdatei (OUFILE) wenn die Daten auch in der Inputdatei (INFILE) vorkommen, also "copy FLT to OUT if exist in IN" **4**: NEGATIV - Negativer Filter, kopiere nur Daten aus der Inputdatei (INFILE) in die Outputdatei (OUFILE) wenn die Daten nicht in der Filterdatei (FILTERFILE) vorkommen, also "copy IN to OUT if not exist in FILTER" **5**: OR - Oder, kopiere Daten aus der Inputdatei (INFILE) bzw. der Filterdatei (FILTERFILE) nur einmal in die Outputdatei (OUFILE) wenn die Daten in einer oder beiden Datein vorkommen. Die Datein sollten gleich struktuiriert sein. Wenn in beiden existiert wird nur INFILE genommen **6**: OR-INV - Oder, analog oben, aber wenn in beiden existiert wird nur FILTERFILE genommen **7**: XOR - Exclusiv Oder, kopiere nur Daten aus der Inputdatei (INFILE) bzw. der Filterdatei (FILTERFILE) in die Outputdatei (OUFILE) wenn die Daten nur einmal vorkommen **8**: JOIN - die beiden Tabellen "joinen", also jeden Satz des IN-File mit jedem Satz des FILTER kombinieren und in Ausgabe übernehmen falls Filter-Spalte übereinstimmen **9**: JOIN-Left - die beiden Tabellen "left outer joinen", also jeden Satz des IN-File mit jedem Satz des FILTER kombinieren und in Ausgabe übernehmen falls Filter-Spalte übereinstimmen oder in FILTER nicht gefunden **10**: JOIN-Right - die beiden Tabellen "right outer joinen", also jeden Satz des IN-File mit jedem Satz des FILTER kombinieren und in Ausgabe übernehmen falls Filter-Spalte übereinstimmen oder in INPUT nicht gefunden **11**: JOIN-Full - die beiden Tabellen "right outer joinen", also jeden Satz des IN-File mit jedem Satz des FILTER kombinieren und in Ausgabe übernehmen falls Filter-Spalte übereinstimmen oder in INPUT oder in FILTER nicht gefunden
`Flt_Col_FLT`	Angabe der Vergleichsspalte für die Filterdatei (FILTERFILE). Zur Ermittlung ob Daten aus der Inputdatei (INFILE) in der Filterdatei (FILTERFILE) existieren wird die ganze Zeile oder bestimmte Spalten verglichen.: **0**: ZEILE - die ganze Zeile wird zum Vergleich herangezogen, sinnvoll insbesondere wenn nur ein Spalte vorhanden **1-N**: SPALTE - die entsprechende X. Spalte in der Datei wird zum Vergleich benutzt
`Flt_Col_IN`	Angabe der Vergleichsspalte für die Inputdatei (INFILE): **0**: ZEILE - die ganze Zeile wird zum Vergleich herangezogen, sinnvoll insbesondere wenn nur ein Spalte vorhanden **1-N**: SPALTE - die entsprechende X. Spalte in der Datei wird zum Vergleich benutzt
`FilterFile`	Datei mit Überschrift und Datenzeilen wenn `FilterArt` > 0, (kein Default, i.d.R. LEER).
`FilterDistinct`	Sollen doppelt auftretende Zeilen (bezüglich Vergleichskriterium) beim Filter speziell behandelt werden: **0**: NORMAL - ggf. duplicate - nichts tun (Default) **1**: DISTINCT - übergehe Doubletten
`FilterNumeric`	Wie sollen die Vergleichskriterien (bezüglich angenommener Sortierung und Vergleichsoperation) behandelt werden: **0**: Nein, nicht numerisch sondern TEXT - als Zeichenkette, die Inhalte der Filterspalte(n) sind i.d.R. gleich lang, Sortierfolge muss 1,2,20,21,3 sein (Default) **1**: Ja, möglichst numerisch als ZAHL - die Inhalte der Filterspalte(n) werden falls reine Zahl numerisch verglichen,

Beispiel:

Ziel: Ermittlung von Milchviehbetrieben ohne Bestandsmilchuntersuchungen

Set 1: Abfrage Betriebe mit BHV1-Bestandsmilchergebnisse

Set 2: Abfrage Betriebe mit Produktionsrichtung Milchviehhaltung

Set 3: Filtere Milchviehbetriebe gegen Betriebe mit BHV1-Bestandsmilchuntersuchungen

HITBatch.INI

```
[SET-1]
MELDUNG=BHV1BEST
COMMAND=RS
DATALINESSOFAR=0
INPUTAFTERSUCCESS=0
TESTDOPPELTEHEADER=1
INFILE=BHV1BEST.in
LOGFILE=BHV1BEST.Log
GOODFILE=BHV1BEST.god
OUFILE=BHV1BEST.out
CSVIN= 0
CSVOUT= 0
VERHALTENBEINACHFRAGE=2
```

Wichtig: Bei Filterfunktionen müssen die Daten gleich sortiert sein; in diesem Bsp. nach Betriebsnummer ORDER;BNR15

HitBatch.ini | BHV1BEST.in | PROD_RICHT.in

```
1 BNR15;UNTS_NDAT;UNTSERG1_B;ANZ_TIERE
2 BNR15;BW;050000000000;059999999999;AND;UNTS_UDAT;BW;01.01.2000;06.06.2012;ORDER;BNR15
```

HitBatch.ini | BHV1BEST.in | PROD_RICHT.in | BHV1BEST.out

```
1 BNR15;UNTS_NDAT;UNTSERG1_B;Anz_Tiere
2 276050000000001;25.04.2011;1;25
3 276051311400001;01.06.2011;0;3
4 276051311400001;01.06.2011;1;1
5 276051311400001;01.06.2011;2;2
6 276051311400001;01.06.2011;4;4
```

```
[SET-2]
MELDUNG=PROD_RICHT
COMMAND=RS
DATALINESSOFAR=0
INPUTAFTERSUCCESS=0
TESTDOPPELTEHEADER=1
INFILE=PROD_RICHT.in
LOGFILE=PROD_RICHT.Log
GOODFILE=PROD_RICHT.god
OUFILE=PROD_RICHT.out
CSVIN= 0
CSVOUT= 0
VERHALTENBEINACHFRAGE=2
```

HitBatch.ini | BHV1BEST.in | PROD_RICHT.in

```
1 BNR15;PROD_VON;PROD_BIS;PROD_S_MIK
2 BNR15;BW;050000000000;059999999999;AND;PROD_BIS;=;31.12.2100;AND;PROD_S_MIK;EQ;1;ORDER;BNR15
```

HitBatch.ini | BHV1BEST.in | PROD_RICHT.in | BHV1BEST.out | PROD_RICHT.out

```
1 BNR15;PROD_VON;PROD_BIS;PROD_S_MIK
2 276050000000001;01.07.2008/00.00.00.0;31.12.2100/00.00.00.0;1
3 276050000000002;01.12.2009/00.00.00.0;31.12.2100/00.00.00.0;1
4 276051311400001;12.02.2009/00.00.00.0;31.12.2100/00.00.00.0;1
```

```
[SET-3]
MELDUNG=*
COMMAND=*
DATALINESSOFAR=0
INPUTAFTERSUCCESS=0
VERHALTENBEINACHFRAGE=1
FILTERART=4
FLT_COL_FLT=1
FLT_COL_IN=1
FILTERFILE=BHV1BEST.out
INFILE=PROD_RICHT.out
OUFILE=Tankmilchen_nicht_vorhanden3.out
LOGFILE=Tankmilchen_nicht_vorhanden3.Log
OUTAPPEND=0
LOGAPPEND=0
CSVIN=0
CSVOUT=0
LOGOUT=0
```

Filter

Die Rinderhalter aus PROD_RICHT (INFILE) werden nun gegen die Betriebe mit BHV1-Untersuchungen abgeglichen (FILTERFILE)
Das Ergebnis sind Milchviehbetriebe ohne BHV1-Bestandsmilchuntersuchungen im OUFILE.
FILTERART=4→ Negativer Filter, kopiere nur Daten aus der Inputdatei (INFILE) in die Outputdatei (OUFILE), wenn die Daten nicht in der Filterdatei (FILTERFILE) vorkommen

3.3.4. Verschiedenes

3.3.4.1. Ausgabe von Land und Kreis als Funktion

```
#ALLWAYS_BW 0 1
*0:XS:LOGON/BNR15;PIN;MELD_WG;TIMEOUT;VHMXLINE_F:276099999999999;681687139163;3;1200;1
*1:RS/C:ERSTERF/$BNRLAND(BNR15;1/Land);$BNRLAND(BNR15;2/L2);$BNRLAND(BNR15;3/L3);$BNRKREIS(BNR
15;1/Kreis);$BNRKREIS(BNR15;0/K0);$BNRKREIS(BNR15;1/K1);$BNRKREIS(BNR15;2/K2);$BNRKREIS(BNR15;
3/K3);$COUNTD(BNR15/Anzahl_Betriebe);$COUNT(/Anzahl_Tiere):BNR15;BW;276091610000000;2760918800
00000;GROUP;$BNRLAND(BNR15);$BNRKREIS(BNR15)
#STOP
```

Ausgabe:

```
;    Land;L2;L3;Kreis;K0;K1;K2;K3;Anzahl_Betriebe;Anzahl_Tiere;
;    9;BY;Bayern;9161;161;9161;IN;Ingolstadt, Stadt;10;68;
;    9;BY;Bayern;9162;162;9162;M;München, Stadt;1;1;
;    9;BY;Bayern;9181;181;9181;LL;Landsberg a. Lech;2;126;
;    9;BY;Bayern;9183;183;9183;MÜ;Mühldorf a. Inn;60;2280;
```

3.3.4.2. Umbenennen von Spalten

Beispiel: *2:RS/C:BTR_D/$RENAME(BNR15/**Betrieb**);**NAME**:BNR15;=;276091630000011

Ausgabe:

```
;    Betrieb;NAME;
;    276091630000011;Rinderhalter Tübingen;
```

Doku dazu unter:

www.hi-tier.de/entwicklung/Konzept/HITP/feink045.html#BNRKREIS

3.3.4.3. Datumsdifferenz im HIT-Protokoll

```
#LINE_NUMBERS 0
*2:RS:CC_VRI/$BNRREGBEZ(BNR15/Reg.Bez);$BNRREGBEZ(BNR15;3/Bezirk);VOK_ANKD;$CODETEXT(VOK_ANKD/
Ankündigung);$DATE_DIFF(VOK_ADAT;VOK_DAT/Ankünd.Differenz);$COUNTD(BNR15/Betriebe);$COUNT(/Kon
trollen):BNR15;BW;09 100 000 0000;09 799 999
9999;AND;VOK_DAT;BW;1.1.2000;31.12.2012;GROUP;$BNRREGBEZ(BNR15);VOK_ANKD;$DATE_DIFF(VOK_ADAT)
#PLAUSI_EXIST_ONLY 121
#STOP

Reg.Bez;Bezirk;VOK_ANKD;Ankündigung;Ankünd.Differenz;Betriebe;Kontrollen;
1;Oberbayern;0;Ohne Ankündigung;%--;32;95; 1;Oberbayern;1;am Prüfungstag;%--;1;1;
1;Oberbayern;2;am Vortag;%--;8;10; 1;Oberbayern;3;innerhalb von 48 h vorher;%--;1;1;
1;Oberbayern;4;noch früher;3;1;1; 4;Mittelfranken;3;innerhalb von 48 h vorher;%--;1;1;
7;Schwaben;0;Ohne Ankündigung;%--;1;2; 7;Schwaben;2;am Vortag;%--;1;1;
```

4. Befehlserstellung - Hilfe über Meldeprogramm

Hilfreich bei Erstellung eigener Abfragen ist, zunächst die Abfrage online durchzuführen, weil die Befehle mit protokolliert werden; sie sind im Meldeprogramm einsehbar unter der Rubrik „Rinderdatenbank-Abfragen, „Weitere Abfragen und Funktionen“ , „**Anzeige der zuletzt durchgeführten internen Befehlen“.**

