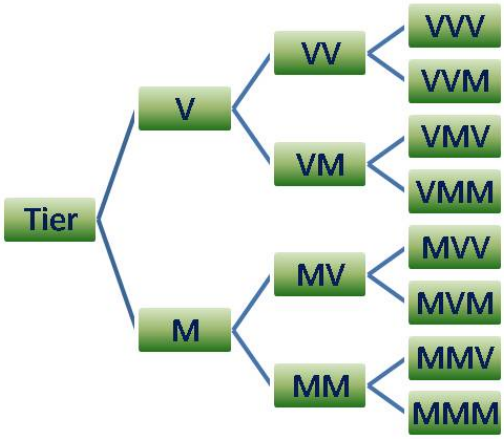
QUALITAS

Übersicht

Existierende Pedigree-Exporte in ARGUS RV1763



Peter von Rohr
FB EDV, Qualitas AG
Chamerstrasse 56, CH-6300 Zug
http://www.qualitasag.ch
peter.vonrohr@qualitasag.ch

Contents

Status des Dokuments

Version	Date	Author	Status	Project
0.0.901	2016-07-25	pvr	Erstellung	RV1763
0.0.902	2016-07-28	pvr	pa_zws_pedi	RV1763

Abkürzungen

Abbreviation	Meaning
ARGUS	Informationssystem für BrunaNet, redonline+, beefnet, etc
BVCH	Braunvieh Schweiz

Erklärung

Dieses Dokument gibt eine Übersicht über bestehende Routinen in `Informationssystem für BrunaNet, redonline+, beefnet, etc` (ARGUS), welche Pedigrees exportieren.

Einführung

Pedigree liegen meist als Tabelle mit mindestens drei Spalten vor. In der minimalen Ausführung des Pedigrees steht auf jeder Zeile ein Individuum mit seinen Eltern. Die Elterninformation muss nicht vollständig sein. Jedes Individuum darf im Pedigree nur einmal vorkommen. Eltern können mehrmals vorkommen.

Zusätzlich zur Abstammungsinformation können auch noch weitere Informationen, wie Geschlecht oder Herdebuchrasse oder Blutanteile angegeben sein.

Pedigree-Exporte in ARGUS

Eine Suche nach dem Schlüsselwort "pedigree" über alle SQL-Quelldateien führte zu folgendem Ergebnis. Die folgende Ergebnistabelle wird mit `searchAllSrc` vom R-Package `SimpleProgCodeAnalysis` gemacht. Das Resultat dieser Suche ist im Anhang dieses Dokuments aufgeführt. Die meisten dieser Suchresultate sind nicht massgebend für die Pedigree Exporte. Es finden sich aber darin gewisse Hinweise, welche Routinen wir genauer untersuchen sollten.

Prozedur pa_zws_pedi.pedigree_rrtdm

Die Prozedur `pedigree_rrtdm` im package `pa_zws_pedi` ist die wahrscheinlich aktuellste Routine für den Export eines Pedigrees. Diese Prozedur schreibt ein Pedigree in eine Datei. Der Export des Pedigrees wird direkt ab der Tabelle `animal` gemacht. Per default lautet der Name der Outputdatei `<yyyymmdd>_pedigree_rrtdm.dat`, wobei `<yyyymmdd>` für das aktuelle Systemdatum steht . In Abhängigkeit der Prozedurparameter `pnExportBlood` und `pnExportNameDame` werden auch Blutanteile der Tiere und Namen von weiblichen Tieren in separate Dateien geschrieben. Die Namen dieser Dateien sind mit den Zusätzen `_Blood_` und `_NameDame_` versehen.

Das Format des exportierten Pedigrees ist in der nachfolgenden Tabelle dargestellt.

Kolonne	StartPosition	EndPosition	Was
1	1	10	TierId
2	12	21	VaterId
3	23	32	MutterId
4	34	37	Geburtsjahr
5	39	40	Rassecode
6	41	43	ITB-Land
7	44	44	ITB-Geschlecht
8	45	56	ITB-Nummer
9	57	70	TVD-Nummer
10	71	78	Geburtsdatum
11	79	81	Rasse(code)
12	82	82	Status
13	84	84	HB-Status

Welche Tiere in einem Export in die Pedigree-Outputdatei geschrieben werden ist abhängig vom Mandanten, für welchen das Pedigree exportiert wird. Der Mandant wird durch das Argument `pnMandant` bestimmt

und nimmt einen Defaultwert von 1 an, d.h. per default wird ein Pedigree für den Mandanten `Braunvieh Schweiz` (BVCH) exportiert. In Abhängigkeit des Mandanten wird ein Set von Rassen definiert. Alle Tiere, welche einer HB-Rasse (Feld `ani_race_id`) angehören, welche in diesem Set ist, werden exportiert.

Pseudotiere sind Tiere, welche im Feld `ANI_PSEUDOTIER` einen Wert von 1 haben, werden beim Export ignoriert. In Abhängigkeit des Prozedurarguments `pnIgnoreOKCode` werden Tiere, welche den OK-Code nicht auf OK gesetzt haben ohne Abstammung, d.h. die Eltern werden auf unbekannt (0) gesetzt, exportiert.

Weitere Exportroutinen in pa_zws_pedi

Die übrigen Exportroutinen in Package `pa_zws_pedi` scheinen alle auf der Hilfstabelle `TH_PEDIGREE` basieren. Das heisst vor dem Export muss zuerst die Hilfstabelle aktualisiert werden. Dies geschieht mit der Funktion `nInsertPedigree(pnAnimalID IN NUMBER, pnFlag IN NUMBER)`. In dieser Funktion wird für ein einzelnes Tier `pnAnimalID` überprüft, ob es schon in der Hilfstabelle vorhanden ist oder nicht. Falls nein, wird das Tier in die Hilfstabelle eingetragen.

Basierend auf der Namensgebung, scheinen die Exportroutinen scheinen spezifisch für verschiedene Merkmale zu sein. So tauchen Bezeichnungen, wie `ttm`, `LBE` oder `ND` auf.

Pedgree Export ab GUI

Der Job `SNP-Pedigree` Export ruft die Prozedur `PA_ZWS_PEDI.pedgree_rrtdm` auf.

Anhang

- Number of occurrences found in search: 26

Table 4: Liste der Funktionsaufrufe in Datei ani_pgb.sql

Zeilennummer	Funktionsaufruf
2959	/* Pedigree-Tiere müssen als NH-Tiere erfasst werd
2960	IF nIdMembre IN (PA_CONST.STATUS_PEDIGREEBTR) THEN

Table 5: Liste der Funktionsaufrufe in Datei bta_jli_pgb.sql

Zeilennummer	Funktionsaufruf
2003	/* Ist der Betrieb ein HB- oder Pedigree-Betrieb?
2004	–> Pedigree-Tiere zählen ab 2007 auch zu den HB-T
2008	,PA_CONST.STATUS_PEDIGREEBTR) THEN
2647	PA_CONST.STATUS_PEDIGREEBTR)
2953	PA_CONST.STATUS_PEDIGREEBTR)
5892	–IDMEMBRE <= 4 AND – Test, Tradition, Basic, Ped
5909	–IDMEMBRE <= 4 AND – Test, Tradition, Basic, Ped
5940	–IDMEMBRE <= 4 AND – Test, Tradition, Basic, Ped

Table 6: Liste der Funktionsaufrufe in Datei bta_lib_pgb.sql

Zeilennummer	Funktionsaufruf
5527	2 = Job für Pedigree
5542	WHEN 2 THEN BTAStartPedigree(pnBTRNr);
5621	WHEN 2 THEN BTAStartOutsourcingSHB(pnBtrNr); –> J
6542	AND (NOT PA_DL.bCheckService(nAdrId, PA_CONST.SERV
6622	AND (NOT PA_DL.bCheckService(nAdrId, PA_CONST.SERV
7454	PROCEDURE BTAStartPedigree(pnBtrId IN NUMBER DEFAU
7545	PA_MEL.SendPipe(sPipe, 'Start BTA-Pedigree-Verarbe
7546	nVertragCode := PA_CONST.STATUS_PEDIGREEBTR;

Zeilennummer	Funktionsaufruf
7622	IF rowBNBtrfuerDruck.IDMEMBRE IN (PA_CONST.STATUS_
7657	PA_MEL.LogInfo(PA_CONST.PROZESS_BTA, 6, '5 Aufbere
7671	–> Bei Pedigree-Betrieben Brief drucken, bei ande
7672	IF (rowBNBtrfuerDruck.IDMEMBRE = PA_CONST.STATUS_P
7674	sText := '[Doctype]BriefPedigree';
7724	PA_MEL.SendPipe(sPipe, 'BTAPedigree-Verarbeitung a

Table 7: Liste der Funktionsaufrufe in Datei btr_pgb.sql

Zeilennummer	Funktionsaufruf
2869	–> Wir setzen nur das neue Wägedatum auf dem Betr
2935	ELSIF PA_DL.bCheckService(nAdrId, PA_CONST.SERVICE

Table 8: Liste der Funktionsaufrufe in Datei capra_doku_pgb.sql

Zeilennummer	Funktionsaufruf
457	PA_SYS.PUT_LINE(F_File,'[ABSTAMMUNG]'||sGetText('c

Table 9: Liste der Funktionsaufrufe in Datei capra_ZWS_pgb.sql

Zeilennummer	Funktionsaufruf
113	–a) Datei Stammdaten Pedigree

Table 10: Liste der Funktionsaufrufe in Datei dsch_exp_pgb.sql

Zeilennummer	Funktionsaufruf
4773	/* Beschreibung : Liefert den Y03 Datensatz für di

Table 11: Liste der Funktionsaufrufe in Datei dsch_imp_3_pgb.sql

Zeilennummer	Funktionsaufruf
2550	(PA_BTR.nGetMembreId(prowParam.nStandortID) = PA_C
2552	–> Pedigreebetriebe bekommen für Braune immer AA
2561	ELSE –> alle Btr ausser Beef und Pedigree

Table 12: Liste der Funktionsaufrufe in Datei dsch_imp_pgb.sql

Zeilennummer	Funktionsaufruf
9981	PROCEDURE ImportItbPedigree200(psFileName IN VARCH
10071	IF (psFileName NOT LIKE 'ImportItbPedigree%'
10076	,psMessage => 'Falscher Filename!'||psFileName||'

Table 13: Liste der Funktionsaufrufe in Datei dsch_kunden_pgb.sql

Zeilennummer	Funktionsaufruf
3301	* @throws EXC_VALIDATION Falls Pedigree-Loo
3324	, psMessage => 'Pedigree-Loop bei Tier;' || to_cha

Table 14: Liste der Funktionsaufrufe in Datei hb_pgb.sql

Zeilennummer	Funktionsaufruf
532	ELSIF PA_BTR.nGetBtrStatus(nBtrID) = PA_CONST.STAT
890	ELSIF PA_BTR.nGetBtrStatus(nBtrID) = PA_CONST.STAT
3414	ELSIF (nStandortID IS NOT NULL) AND (PA_BTR.nGetMe
3415	–> Pedigreebetriebe bekommen für Braune immer AA
3424	ELSE –> alle Btr ausser Beef und Pedigree
5269	–> wir müssen Pedigree- und Beef-Betriebe von den
5270	IF rowAni.BTRSTATUS = PA_CONST.STATUS_PEDIGREEBTR
5271	–> Bei Pedigree müssen wir MRAW und AA setzen
5275	sArtCode := PA_CONST.ARTGRP_DOKU_AA_PEDIGREE;
9470	FOR cRacePedigree IN (SELECT a.ANI_RACE_ID A_A
9479	IF (cRacePedigree.A_ANI_OK_ID <> nOK)
9480	OR (cRacePedigree.V_ANI_OK_ID <> nOK)
9481	OR (cRacePedigree.M_ANI_OK_ID <> nOK) THEN
9485	IF (NOT aAllowedRace.EXISTS(cRacePedigree.A_AN
9486	OR (NOT aAllowedRace.EXISTS(cRacePedigree.A_ANI_RA
9487	OR (NOT aAllowedRace.EXISTS(cRacePedigree.A_ANI_RA

Table 15: Liste der Funktionsaufrufe in Datei laktstat_2_pgb.sql

Zeilennummer	Funktionsaufruf
1285	AND HFAEXPL.HFAEXPLDATECL IS NOT NULL –

Table 16: Liste der Funktionsaufrufe in Datei medi_pgb.sql

Zeilennummer	Funktionsaufruf
911	– BVCH: Behandlungen von Pedigree- / Beef-Betrieb
920	AND SADR_SERVICE_ID I
923	sComment := 'Beef/Pedigree-Betrieb;'||pa_btr.sBtrN

Table 17: Liste der Funktionsaufrufe in Datei mlp_mpv_pgb.sql

Zeilennummer	Funktionsaufruf
4393	(rowRind.INSDATE > TRUNC(SYSDATE-330)) THEN –> Pr

Table 18: Liste der Funktionsaufrufe in Datei shared_doku_2_pgb.sql

Zeilennummer	Funktionsaufruf
1018	– bei BVCH mit Dienstleistung "Bruna-Test", "Brun
1052	,PA_CONST.STATUS_PEDIGREEBTR)
1176	,PA_CONST.STATUS_PEDIGREEBTR)
1927	WHERE FTRA_ART_ID IN (10521,10795)
2004	/* Fuer BVCH: Ist der Betrieb ein HB- oder Pedigre
2008	,PA_CONST.STATUS_PEDIGREEBTR)

Table 19: Liste der Funktionsaufrufe in Datei sheep_doku_pgb.sql

Zeilennummer	Funktionsaufruf
343	PA_SYS.PUT_LINE(F_File,'[ABSTAMMUNG]'||sGetText('s
361	PA_SYS.PUT_LINE(F_File,'[ABSTAMMUNG]'||sGetText('s

Table 20: Liste der Funktionsaufrufe in Datei snp_pgb.sql

Zeilennummer	Funktionsaufruf
3799	EXECUTE IMMEDIATE ‘TRUNCATE TABLE TH_SNP_PEDIGREE’
3857	FROM T_SNPINFO, TH_SNP_PEDIGREE, ANIMAL
3922	INSERT INTO TH_SNP_PEDIGREE (SPEDI_ANIMAL_ID) VALU
4174	IF aFeald(29) != ‘Pedigreeimputiert’ THEN

Table 21: Liste der Funktionsaufrufe in Datei stat_pgb.sql

Zeilennummer	Funktionsaufruf
65	–2. Anzahl lebende BV-Kühe ohne Pedigree
84	UTL_FILE.PUT_LINE(fOutput, ’Anzahl lebende BV-Kühe
116	–4. Anzahl lebende OB-Kühe ohne Pedigree
135	UTL_FILE.PUT_LINE(fOutput, ’Anzahl lebende OB-Kühe

Table 22: Liste der Funktionsaufrufe in Datei vzg_pgb.sql

Zeilennummer	Funktionsaufruf
299	nArtikelPedigreeId NUMBER := 10795;
389	,PA_CONST.STATUS_PEDIGREEBTR) AND
444	FTRA_ART_ID = nArtikelPedigreeId) AND
724	nArtikelPedigreeId NUMBER := 10795;
754	T_FAKTTRANSFER.FTRA_ART_ID IN (nArtikelId,nArtikel

Table 23: Liste der Funktionsaufrufe in Datei zws_fbk_pgb.sql

Zeilennummer	Funktionsaufruf
97	EXECUTE IMMEDIATE ‘TRUNCATE TABLE TH_PEDIGREE’;
135	– Auffüllen der TH_PEDIGREE (Eltern, Grosseltern)
138	– PA_MEL.SendPipe(sPipe,’Start ZWS-FBK sel
142	– /* Export TH_PEDIGREE in pedig_ascii_lbe
143	– PA_MEL.SendPipe(sPipe,’Start ZWS-FBK ped
145	– PA_ZWS_PEDI.pedigree_ttm(’pedigree_fbk.d
224	,TH_PEDIGREE

Zeilennummer	Funktionsaufruf
226	AND ANIMAL.IDANIMAL =
	TH_PEDIGREE.PEDI_ANIMAL_ID(+
229	AND TH_PEDIGREE.PEDI_ANIMAL_ID IS NULL) LOOP
262	/* Tier darf nicht bereits in TH_PEDIGREE sein, so
263	IF PA_ZWS_PEDI.nInsertPedigree(rowTier.IDANIMAL,0)
264	PA_MEL.LogFehler(PA_CONST.PROZESS_ZWS_FBK,1,'Tier
496	<>
499	FROM TH_PEDIGREE
503	END LOOP PedigreeSexLoop;
508	<>
511	FROM TH_PEDIGREE
516	END LOOP PedigreeFlagLoop;

Table 24: Liste der Funktionsaufrufe in Datei zws_gal_pgb.sql

Zeilennummer	Funktionsaufruf
63	– File mit Pedigree generieren
64	– PA_MEL.SendPipe(sPipe,'Start
	pedigree ascii-Dat
65	– PA_ZWS_PEDI.PEDIGREE_RRTDM;

Table 25: Liste der Funktionsaufrufe in Datei zws_lbe_pgb.sql

Zeilennummer	Funktionsaufruf
33	/* Beschreibung : Initialisiert die TH_PEDIGREE un
47	EXECUTE IMMEDIATE 'TRUNCATE TABLE TH_PEDIGREE';
60	PA_MEL.SendPipe(sPipe,'TH_PEDIGREE initialiseirt u
75	/* den Export der Pedigree-Daten
112	/* Auffüllen der TH_PEDIGREE (Eltern, Grosseltern)
116	PA_MEL.SendPipe(sPipe,'Start ZWS-LBE selAllAnimals
121	/* Codierung setzen auf TH_PEDIGREE */
130	/* Export TH_PEDIGREE in pedig_ascii_lbe.dat */
321	, ANIMAL.IDANIMAL IDANIMAL
322	, ANIMAL.IDPERE
323	, ANIMAL.IDMERE
430	,TH_PEDIGREE
579	/* Hier wird das Tier in die Pedigree-Tabelle gefü
585	PA_MEL.LogFehler(PA_CONST.PROZESS_ZWS_LBE,1,'Tier
630	,TH_PEDIGREE
633	AND DLC.DLC_ANIMAL_ID =
	TH_PEDIGREE.PEDI_ANIMAL_ID
635	AND (TH_PEDIGREE.PEDI_ANIMAL_ID IS NULL
636	OR TH_PEDIGREE.PEDI_FLAG >2)
2337	FROM TH_PEDIGREE
2347	FROM TH_PEDIGREE
2360	FROM TH_PEDIGREE
2366	FROM TH_PEDIGREE
2372	FROM TH_PEDIGREE

Zeilennummer	Funktionsaufruf
2378	FROM TH_PEDIGREE
2381	FROM TH_PEDIGREE P
2382	WHERE P.PEDI_ANIMAL_ID = TH_PEDIGREE.PEDI_VATER_ID
2383	PA_SYS.PUT_LINE(fFile,'Anzahl Väter, die nicht im
2386	FROM TH_PEDIGREE
2389	FROM TH_PEDIGREE P
2390	WHERE P.PEDI_ANIMAL_ID = TH_PEDIGREE.PEDI_MUTTER_I
2391	PA_SYS.PUT_LINE(fFile,'Anzahl Mütter, die nicht im
3162	PA_MEL.SendPipe(sPipe,'TH_PEDIGREE initialiseirt u
3224	FROM dlc, TH_PEDIGREE
3226	AND TH_PEDIGREE.PEDI_ANIMAL_ID IS NULL
3371	PA_MEL.LogFehler(PA_CONST.PROZESS_ZWS_LBE,1,'Tier

Table 26: Liste der Funktionsaufrufe in Datei zws_muku_pgb.sql

Zeilennummer	Funktionsaufruf
582	* Export Abstammungen (Pedigree) aller für Mutterk
596	PROCEDURE ZwsExportPedigree IS
615	, psFileName => 'pedigree_muku_'||TO_CHAR(SYSDATE
622	<>
733	EXIT Pedigree_LOOP;
737	END LOOP Pedigree_LOOP;
749	END ZwsExportPedigree;

Table 27: Liste der Funktionsaufrufe in Datei zws_nd_pgb.sql

Zeilennummer	Funktionsaufruf
40	/* Beschreibung : Initialisiert die TH_PEDIGREE un
51	EXECUTE IMMEDIATE 'TRUNCATE TABLE TH_PEDIGREE';
53	PA_MEL.SendPipe(sPipe,'TH_PEDIGREE initialisiert u
73	– – Tabelle T_PEDIGREE leeren (=Truncate)
83	– – Pedigree auffüllen mit Vorfahren
84	– PA_MEL.SendPipe(sPipe,'Pedigree auffülle
89	– PA_MEL.SendPipe(sPipe,'Pedigree-File ers
90	– pa_zws_pedi.pedigree_nd;
111	/* 22.02.2008 * BM * Pedigree etc nicht mer n
122	– Tabelle T_PEDIGREE leeren (=Truncate)
132	– Pedigree auffüllen mit Vorfahren
133	– PA_MEL.SendPipe(sPipe,'Pedigree auffülle
138	– PA_MEL.SendPipe(sPipe,'Pedigree-File ers
139	– pa_zws_pedi.pedigree_nd;
209	,TH_PEDIGREE
211	AND ANIMAL.IDANIMAL = TH_PEDIGREE.PEDI_ANIMAL_ID(+
212	AND TH_PEDIGREE.PEDI_ANIMAL_ID IS NULL

Zeilennummer	Funktionsaufruf
489	PA_ZWS_PEDI.ins_animals(rowAnimal.IDANIMAL,CONST_P
552	,TH_PEDIGREE
554	AND ANIMAL.IDANIMAL = TH_PEDIGREE.PEDI_ANIMAL_ID(+
555	AND TH_PEDIGREE.PEDI_ANIMAL_ID IS NULL
1476	/* Beschreibung : erstellt File ab Pedigree für Ab
1489	FROM TH_PEDIGREE
1531	PA_SYS.PUT_LINE(fFile,'Anzahl Je Sex-Code im Pedig
1535	FROM TH_PEDIGREE
1541	PA_SYS.PUT_LINE(fFile,'Anzahl Je Flag im Pedigree:
1545	FROM TH_PEDIGREE

Table 28: Liste der Funktionsaufrufe in Datei zws_pedi_pgb.sql

Zeilennummer	Funktionsaufruf
6	/* Beschreibung : Package Body, erstellt Pedigree
53	– FROM TH_PEDIGREE;
57	– FOR curSelAnimal IN (SELECT * FROM th_pe
127	– FROM TH_PEDIGREE;
150	nTemp := nInsertPedigree(pnAnimalID,pnFlag);
153	FUNCTION nInsertPedigree(pnAnimalID IN NUMBER, pnF
155	/* Name : nInsertPedigree
174	SELECT COUNT(*) INTO nAnimalCounter FROM th_pedigr
218	INSERT INTO TH_PEDIGREE(PEDI_ANIMAL_ID,
270	FROM TH_PEDIGREE;
271	SP_SYS_CREATE_SEQUENCE_START('S_PEDIGREECODE', nCo
273	FROM TH_PEDIGREE
277	curSetCode.PEDI_CODIERUNGTIER := PA_SYS.nGetSequen
278	UPDATE TH_PEDIGREE
290	FROM TH_PEDIGREE;
295	FROM TH_PEDIGREE
310	UPDATE TH_PEDIGREE
324	/* Name : PEDIGREE_TTM
325	/* Beschreibung : erstellt file PEDIGREE_TTM.DAT
341	/* File f. Pedigree-Daten öffnen */
344	FOR curSelPedi IN (SELECT * FROM TH_PEDIGREE) LOOP
370	PA_MEL.SendPipe(sPipe,'Fehler bei pedigree_ttm, Ti
374	/* File PEDIGREE_ND.DAT schliessen */
377	END PEDIGREE_TTM;
403	FOR curSelPedi IN (SELECT * FROM TH_PEDIGREE ORDER
445	/* Name : PEDIGREE_ND
446	/* Beschreibung : erstellt file PEDIGREE_ND.DAT
460	/* File PEDIGREE_ND.DAT öffnen */
461	fFileHandle := UTL_FILE.FOPEN(GESTHO.PA_CONST.FILE
464	FROM TH_PEDIGREE
476	/* File PEDIGREE_ND.DAT schliessen */

Zeilennummer	Funktionsaufruf
479	END PEDIGREE_ND;
574	/* Name : PEDIGREE_RRTDM
575	/* Beschreibung : erstellt file PEDIGREE_RRTDM.DA
638	/* File f. Pedigree-Daten öffnen */
826	PA_MEL.SendPipe(sPipe,'Fehler bei pedigree_ttm, Ti
842	END PEDIGREE_RRTDM;

Table 29: Liste der Funktionsaufrufe in Datei zws_rrtdm_pgb.sql

Zeilennummer	Funktionsaufruf
158	PA_MEL.SendPipe(sPipe,'pedigree ascii-Datei separa
159	– PA_MEL.SendPipe(sPipe,'Start pedigree asci
160	– PA_ZWS_PEDI.pedigree_rrtdm;
162	– WriteStat(fLogFileHandle,'pedigree_rrtdm',
867	– Tier in Pedigreetabelle einfügen –
869	– tier hat keine Milchproben also nicht in Pedigr