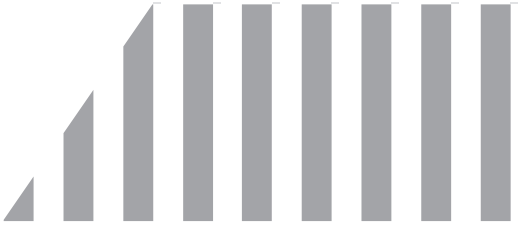
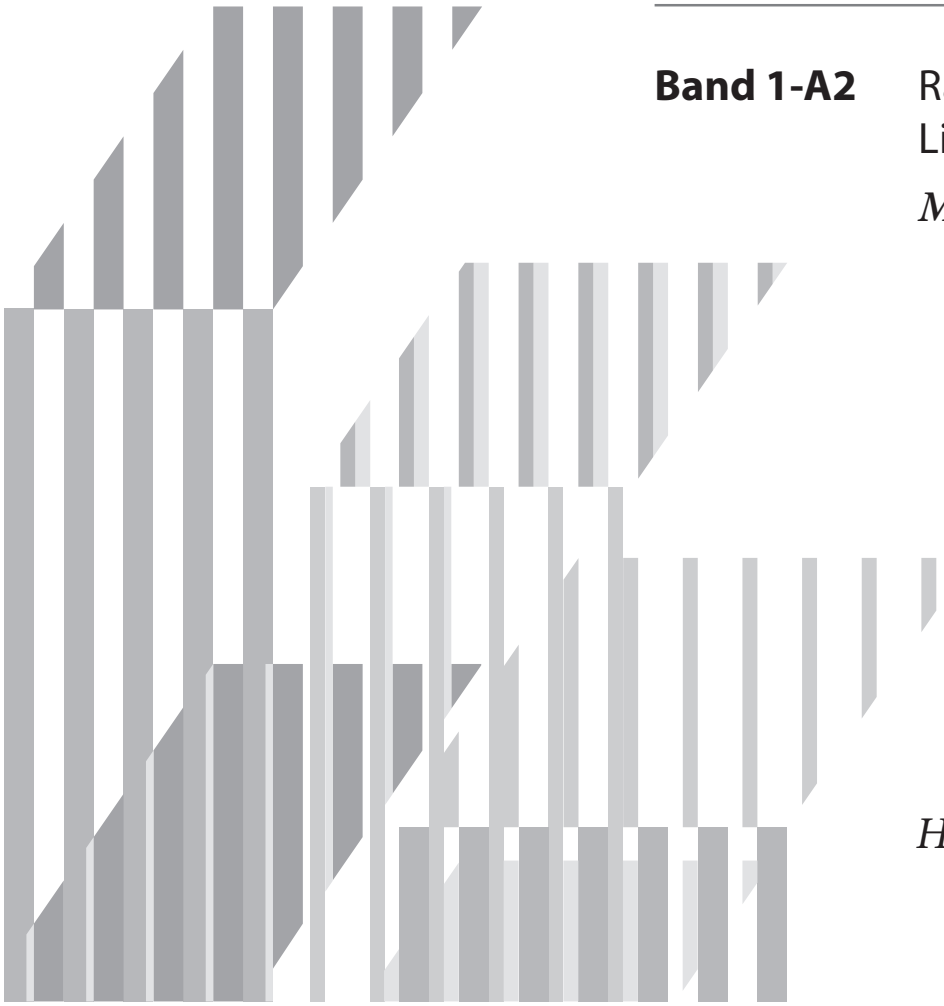
Band 1-A2



MODERNE STRUKTURIERTE PROGRAMMIERUNG

Objektorientiert und algorithmisch
ENTWERFEN | IMPLEMENTIEREN | TESTEN

2. Auflage



Band 1-A2 Ranggruppen
Link-Listen
Methode

Horst van Bremen

Bibliografische Information der Deutschen Nationalbibliothek:
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar.

Gestaltung	Katharina Schmitt
Titel- und Einbandgrafik	Monika Sylvia Gomm † 1971
Englisch-Korrektorat	David Roseveare
Verlag	BoD · Books on Demand GmbH, Überseering 33, 22297 Hamburg, bod@bod.de
Druck	Libri Plureos GmbH, Friedensallee 273, 22763 Hamburg
Version / Datum	Version 2.1.1 vom 1.7.2026
Literaturverzeichnis	siehe Kapitel „18.1 Literaturverzeichnis“ auf Seite 541 f
Rechtliches	siehe Kapitel „18.2 Rechtliche Hinweise“ auf Seite 543 ff
ISBN des Ringbuchs	978-3-6963-6425-0

Über den Autor

Horst van Bremen
Diplom 1972 an der TU Darmstadt – Elektrotechnik und Informatik
39 Berufsjahre in der IT, davon 18 als Freiberufler

IT-Systemarchitekt
Datenbankexperte

Programmiererfahrung seit 1969
Strukturierte Programmierung nach Jackson seit 1974
Freier Autor seit 2011

Vorwort zur ersten A2-Ausgabe

Ranggruppen und Link-Listen – abstrakter geht es kaum. Was unter Laien als eine dieser vielen seltsamen Sprachverirrungen gelten mag, für welche die Informatik berühmt-berüchtigt ist, besitzt dennoch eine hohe praktische Relevanz. Telefonbücher oder Online-Verzeichnisse, Wörterbücher, Inhaltsverzeichnisse, steuerliche Auswertungen, Bundesligatabellen, Bibliotheken-Bestandsregister, Anlagevermögen-Aufstellungen und zahllose weitere Auflistungen basieren auf einem Spektrum von manuellen bis hochautomatisierten Techniken, bei denen es um – eventuell hierarchisch geschachtelte – Reihenfolgen oder Rankings geht.

Sie alle lassen sich mit JSP-Struktogrammen beschreiben und – datentechnisch aufbereitet – in vielfältiger Weise gestalten und verwerten. Allerdings sind diese nützlichen und allgegenwärtigen Listen meistens völlig frei von jeglichem intellektuellen und ästhetischen Reiz, also eine knochentrockene Angelegenheit, die ein Autor weder seinen Leserinnen und Lesern noch sich selbst als interessante Thematik schmackhaft zu machen vermag. Die Fallstudie zum fiktiven Internationalen Schulsportwettbewerb im hier vorliegenden Teil A2 des Bands 1 bildet dagegen sowohl hinsichtlich der Anschaulichkeit als auch der Komplexität eine Ausnahme in dieser Ödnis – so steht es jedenfalls zu hoffen. Letztlich entscheiden das natürlich Sie als Leserin oder Leser.

Die Texte und Grafiken der Fallstudie sind nicht als lineare Abfolge von Lektionen gedacht. Das wäre nicht nur extrem langweilig, sondern würde auch dem Lernziel nicht gerecht, das Schwierigkeitsniveau A2 bei der Beherrschung von JSP/MSP zu erreichen. Das bedeutet, dass hier die Grundlagen für eine selbständige Anwendung der Methode und für deren praktische Umsetzung gelegt werden sollen. Repräsentative Anwendungsfälle für JSP/MSP, beispielsweise in der kommerziellen Programmierung, sind nämlich allenfalls so komplex wie die Fallstudie. Daher behandeln die folgenden Kapitel

- die Abbildung des „Daten-Rohmaterials" in die aufgabenspezifischen Struktogramme,
- deren Umsetzung in die Programmstrukturen,
- die Verifikation dieser Strukturen anhand von Beispielfällen,
- eine inkrementelle algorithmische Implementierung und die zugehörigen Tests,
- die Transformation der JSP/MSP-Entwürfe mit dem Ziel der OO-Implementierung,
- die Anpassung der Basismaschine an die Eigenschaften der verwendeten Java-Klassen,
- den Entwurf einer flexiblen Datei-Zugriffsschicht für Java-Programme und
- das Thema „Test" aus mehreren Perspektiven.

Auch das sind abstrakt erscheinende Themen, aber ihre Umsetzung anhand der Fallstudie ermöglicht es, die methodische Stringenz mit der Anschaulichkeit der Wettbewerbsaspekte zu verbinden. Final geht es darum, das Potential von JSP/MSP auch bei komplexen Aufgaben zu demonstrieren und das Vorgehen zu deren Lösung anhand überschaubarer Beispiele nachvollziehbar darzustellen. Bei dem Schritt, die hier gewonnenen Erkenntnisse in Ihre eigene berufliche Praxis umzusetzen, wünsche ich, der Autor, Ihnen ein gutes Händchen.

Lemgo, den 14.2.2025

Übersichtsverzeichnis Band 1-A2

Inhaltsverzeichnis Band 1-A2

Band 1-A2

06/2015 Römischer Tempel „Maison Carrée“ in Nîmes, Hauptort des Départements Gard (Okzitanien)

Ranggruppenverarbeitung I

Die antike griechische Kultur, insbesondere die aus ihr hervorgehende grandiose Tempelarchitektur, fand in der Mittelmeerwelt viele Bewunderer und Nachahmer. So entstand auch dieser Tempel im griechischen Stil[1] auf dem Forum des antiken Nemausus – Nîmes –, das die Römer 121 v.Chr. eroberten. Den „alten Griechen" wird die Erfindung der kommunalen Demokratie zugeschrieben, wobei allerdings nur ein kleiner – und (einfluss)reicher – Teil der jeweiligen Stadtbevölkerung stimmberechtigt war. Die weltweit bekannteste griechische Errungenschaft ist die der olympischen Wettkämpfe, die 1894 durch Pierre de Frédy, Baron de Coubertin, mit der Gründung des Internationalen Olympischen Komitees wiederbelebt wurden. Auch ausserhalb der olympischen Wettkämpfe hat sich die – in der Antike etablierte – Ehrung der Sieger mit Gold-, Silber- und Bronzemedaillen international durchgesetzt.

Sportwettbewerbe aller Arten haben sich zu den populärsten öffentlichen Ereignissen entwickelt. Im Kleinen sind es vor allem die Schulen, die – zum Beispiel in Deutschland mit den Bundesjugendspielen – an den olympischen Gedanken anknüpfen. Wenn man diese Sportwettbewerbe international generalisiert, dann ergibt sich ganz von selbst ein gutes Beispiel für die Thematik, die im Folgenden behandelt wird.

„Rang" meint eine hierarchische Ordnung, also eine Gliederung der Daten in Gruppen (Rang 1), Untergruppen (Rang 2), Unteruntergruppen (Rang 3) und so weiter. Man kann sich das wie eine russische Matrjoschka-Puppe vorstellen, die bemalt und hohl ist, in der eine etwas kleinere bemalte hohle Puppe steckt, die wiederum eine noch kleinere Puppe enthält, bis ganz innen die kleinste – bemalte, aber nicht hohle – Puppe erreicht ist. Diese Analogie hinkt allerdings, denn wir Programmierer(innen) können zahllose „Puppen" nebeneinander und ineinander schachteln, was mit Holz nicht so gut klappt.

Ähnlich geschachtelte „Puppen" haben wir schon viele gesehen, ohne sie als solche zu begreifen: Es sind die Strukturblöcke der JSP-Struktogramme. Ihre Besteht-aus-Relation passt hervorragend zu einer solchen Aufgabe, wie sie die Ranggruppenverarbeitung ist. Allerdings: Die Beispiele zur Gruppenverarbeitung kommen mit einem einzigen Rang aus und sind daher nur Aufwärmübungen im Vergleich zu realistischen Aufgaben.

Zwei Beispiele aus der Praxis des Autors mit Ranggruppenverarbeitungen seien hier genannt:

- Für ein steuerliches Auswertungssystem waren unter anderem Aufstellungen zu erzeugen, anhand derer die Kunden einer Bank die Erträge ihrer Wertpapiergeschäfte und sonstiger Geldanlagen – also normalerweise Zinserträge – in die dafür vorgesehene Anlage der Steuererklärung übernehmen konnten. Dafür mussten die Einzelerträge über mehrere Stufen hinauf nach Ertragsarten und weiteren Gruppierungsmerkmalen kumuliert werden, um schliesslich zu den Positionssummen für die Steuererklärung zu gelangen.
- Als die Deutsche Telekom die ersten ISDN-Vermittlungsstellen in Betrieb nahm, benötigte sie auch eine neue Abrechnungssoftware, um die auf Bändern gesammelten Verbindungsdaten auszuwerten, die verbrauchten Einheiten zu tarifieren, die so ermittelten Einzelbeträge in einer Datenbank zwischenzuspeichern, an jedem Monatsende daraus für die ISDN-Kunden die Rechnungsbeträge aufzukumulieren, sowie diese den geografisch verteilten Buchungsstellen zuzuordnen und demgemäß aufgesplittet auf etliche Bänder zu schreiben. Der monatliche Lauf war daher als Ranggruppenverarbeitung mit anschliessender Entflechtung der Ergebnisdaten auf die Band-Ausgabedateien gestaltet.

Da das Finanzamt denkbar unbeliebt ist, und Telefonrechnungen auch nicht wesentlich höher angesehen sind, verbietet es sich, damit ein hier verwendbares Beispiel aufzubauen. Ausserdem wäre es in beiden Fällen viel zu speziell, ohne dass dies durch einen signifikanten Lernerfolg gerechtfertigt werden könnte.

Endnoten: siehe Seite 235

Die nachfolgende – sehr ausführliche – **Fallstudie** baut auf dem bisher erreichten Wissensstand auf und erhöht den Schwierigkeitsgrad deutlich, um zu demonstrieren, dass die JSP-Methode weit höheren Ansprüchen genügt, als bisher erkennbar war. Das gilt auch für ihren Nutzen als Basis für algorithmische *und* objektorientierte Implementierungen.

9.1 Der Internationale Schulsportwettbewerb

Zahlreiche in- und ausländische Schulen veranstalten jedes Jahr einen – fiktiven – Sportwettbewerb. Dabei werden die Leistungen in den Sportarten 50-m- oder 100-m-Lauf, Weitsprung und Weitwurf an jeder teilnehmenden Schule und in den Altersstufen ab 10 Jahren in einem bestimmten Monat ermittelt. Es ist jeder Schule freigestellt, ob sie nur einzelne Altersstufen teilnehmen lassen möchte. Aus Gründen der fairen Vergleichbarkeit der durchschnittlichen Leistungen müssen diese Stufengruppen allerdings möglichst vollständig teilnehmen.

Die Ergebnisse werden nach Alter und Geschlecht differenziert mit Punkten von 0 bis 20 bewertet, grob sortiert und – unter Mitgabe der jeweiligen Schul-Identifikation – einer Zentralstelle zugesandt. Diese hat die Aufgabe, diese Daten zu sammeln und auf Vollständigkeit zu prüfen, sie in ein einheitliches Format umzusetzen, zu sortieren und schliesslich darauf basierende Auswertungen zu erstellen. Über die technischen Details der Datenerfassung – beispielsweise mit maschinenlesbaren Erfassungsbogen oder Tabellenkalkulations-Formularen – und die ganzen sonstigen Mühen bis hin zur Bereitstellung der auswertbaren, sortierten Datei mit sämtlichen Sportwettbewerbsdaten sehen wir hier großzügig hinweg. Diese Aspekte tragen zum Beispiel nichts bei.

Dagegen ist es wichtig, wie die Sätze in dieser Datei namens „Sports Dataset“ aufgebaut und sortiert sind. Wegen der gewünschten unproblematischen maschinellen Auswertung haben diese Sätze ein fest definiertes Format, das aber erst weiter unten vorgestellt werden wird. Zunächst sollten wir etwas Anschauung gewinnen, indem wir die Inhalte in einer leicht lesbaren Form zeigen.

9.1.1 Vorläufige Eingabe-Datenstruktur

Für eine einzelne Schule und einen zufällig ausgewählten Teil der Schüler(innen) sehen diese Daten, als Tabelle aufbereitet, beispielweise wie folgt aus:

Liceo Federico Fellini, 92014 Porto Empedocle, Sicilia, Italia

country code	region ID	school #	gender code	age	100m / 50m sprint	long jump	long throw	name
ITA	SI	45008132	F	15	15	16	7	Coglitore, Laura
ITA	SI	45008132	F	15	4	7	6	Peritore, Mariuccia
ITA	SI	45008132	F	16	3	7	6	Di Stefano, Serena
ITA	SI	45008132	F	16	3	9	8	Lo Porto, Mariuccia
ITA	SI	45008132	F	16	18	15	17	Prestìa, Angela
ITA	SI	45008132	F	17	10	8	12	Castagnola, Michela
ITA	SI	45008132	F	17	13	11	15	Clemenza, Ersilia
ITA	SI	45008132	F	17	6	4	7	Fiandaca, Angela
ITA	SI	45008132	F	18	18	15	13	Foti, Antonietta
ITA	SI	45008132	F	18	11	6	8	Memmi, Annaluisa
ITA	SI	45008132	F	18	7	9	5	Panarello, Anna
ITA	SI	45008132	F	18	12	14	16	Testa, Giulia
ITA	SI	45008132	F	18	5	2	4	Tripòdi, Elvira

country code	region ID	school #	gender code	age	100m / 50m sprint	long jump	long throw	name
ITA	SI	45008132	M	13	9	6	9	Ciccione, Pippo
ITA	SI	45008132	M	13	14	12	15	Monteceppo, Lillino
ITA	SI	45008132	M	13	19	17	16	Moscato, Salvo
ITA	SI	45008132	M	13	18	16	14	Sanfilippo, Saverio
ITA	SI	45008132	M	13	17	14	13	Sicumero, Pietro
ITA	SI	45008132	M	13	8	14	11	Verruso, Masino
ITA	SI	45008132	M	14	10	7	15	Bonpensiero, Annibale
ITA	SI	45008132	M	14	9	8	7	Diliberto, Titìllo
ITA	SI	45008132	M	14	17	16	11	Liotta, Pasqualì
ITA	SI	45008132	M	14	7	11	10	Rossi, Angelo
ITA	SI	45008132	M	14	15	16	13	Zaccaria, Gaspare
ITA	SI	45008132	M	15	1	4	5	Ficherà, Cosimo
ITA	SI	45008132	M	15	10	14	9	Landolina, Pasqualino
ITA	SI	45008132	M	15	9	13	17	Ragusano, Saverio
ITA	SI	45008132	M	16	2	3	4	Cosentino, Pippo
ITA	SI	45008132	M	16	0	3	2	Zummo, Antonio
ITA	SI	45008132	M	17	11	10	16	Marchica, Michele
ITA	SI	45008132	M	17	13	5	7	Pisciotta, Giosuè
ITA	SI	45008132	M	18	5	4	8	Comaschi, Tano
ITA	SI	45008132	M	18	16	19	15	Gangitano, Eugenio
ITA	SI	45008132	M	18	12	11	16	Sutera, Nenè

Urhebberrechtlicher **Hinweis**: siehe Endnote [2]

Die englischen Spaltenüberschriften wurden wegen der Bezeichnungen der künftigen Datenelemente gewählt. Die Tabellenzeilen sind nach Land-Code, Regions-ID, Schulnummer, Geschlecht und Alter aufsteigend sortiert. Die ersten drei Werte zusammen identifizieren die Schule, weil Schulnummern nur innerhalb ihrer jeweiligen Region – landesspezifisch – eindeutig sind. Da die Schul-Identifikation hier konstant ist, soll uns eine weitere Tabelle zeigen, wie der Gesamtbestand aufgebaut ist. Dieser kann sehr groß werden. Deshalb sind anschliessend nur Stichproben – wieder mit fiktiven Daten und Namen – gezeigt:

country code	region ID	school #	gender code	age	100m / 50m sprint	long jump	long throw	name
DEU	BW	00470811	F	10	8	9	6	Leibel, Maria
DEU	BW	00470811	F	10	11	8	12	Russo, Antonella
DEU	BW	00470811	...	...	...	...	...	...
DEU	BW	00470811	M	19	12	13	8	Hagelund, Klaus
DEU	BW	00470811	M	19	15	10	11	Pfälzer, Gerhard
DEU	BW	...	...	...	...	...	...	...
DEU	BW	07654321	F	14	10	13	9	Gerlang, Katja
DEU	BW	07654321	F	14	8	5	12	Schroedinger, Katharina
DEU	BW	07654321	...	...	...	...	...	...
DEU	BW	07654321	M	18	7	5	4	Brunzer, Ernst
DEU	BW	07654321	M	18	14	16	17	Biermann, Tobias
DEU	...	...	...	...	...	...	...	...

country code	region ID	school #	gender code	age	100m / 50m sprint	long jump	long throw	name
DEU	NW	00017050	F	16	16	18	14	Lallmann, Anita
DEU	NW	00017050	M	12	11	13	11	Frost, Thomas
DEU	NW	00017050	...	...	...	...	...	...
DEU	NW	00017050	M	17	5	3	6	Starzenbacher, Horst
DEU	NW	00017050	M	17	6	5	6	Waldberg, Peter
...	...	...	...	...	...	...	...	...
ESP	MD	03007877	F	17	9	11	10	Alberto, Carmela
ESP	MD	03007877	F	17	14	13	12	Lavides, Carmela
ESP	MD	03007877	...	...	...	...	...	...
ESP	MD	03007877	M	18	13	17	11	Carmán, Félix
ESP	MD	03007877	M	18	17	18	15	Espacia, Sixto
ESP	MD	03007877	M	18	3	5	4	Gamacha, Fernando
ESP	MD	03007877	M	18	16	15	14	Sertas, José
...	...	...	...	...	...	...	...	...
FRA	IF	00044044	F	13	16	14	12	Frammière, Louise
FRA	IF	00044044	M	11	10	12	14	Duveille, Sylvain
FRA	IF	00044044	M	11	13	11	15	Moustin, Gérard
FRA	IF	00044044	M	11	14	10	11	Mentail, Daniel
...	...	...	...	...	...	...	...	...
ITA	SI	45008132	F	15	15	16	7	Coglitore, Laura
ITA	SI	45008132	F	15	4	7	6	Peritore, Mariuccia
ITA	SI	45008132	...	...	...	...	...	...
ITA	SI	45008132	M	18	16	19	15	Gangitano, Eugenio
ITA	SI	45008132	M	18	12	11	16	Sutera, Nenè

Den Rang 1 bilden die internationalen Land-Codes. Im zweiten Rang stehen die länderspezifischen Regions-IDs. Die regionsspezifischen Nummern der teilnehmenden Schulen stehen auf dem dritten Rang. Der vierte Rang kennt nur zwei Ausprägungen: F = Feminin | M = Maskulin. Die Altersgruppen stellen also den fünften Rang dar.

Es ist unwichtig, ob die Ränge – so wie hier – als Spalten mit steigenden Rangnummern angeordnet sind, oder in irgendeiner anderen Folge. Eine solche Anordnung wird aber nicht selten gewählt, weil dadurch die hierarchischen Beziehungen leicht zu erkennen sind und die Vergabe von Sortierkriterien besonders einfach wird.

Das vorläufige Eingabe-Struktogramm („Abbildung 9.1: Datenstruktur der Sportwettbewerbs-Eingabedaten“ auf Seite 195) beschreibt diese Daten.

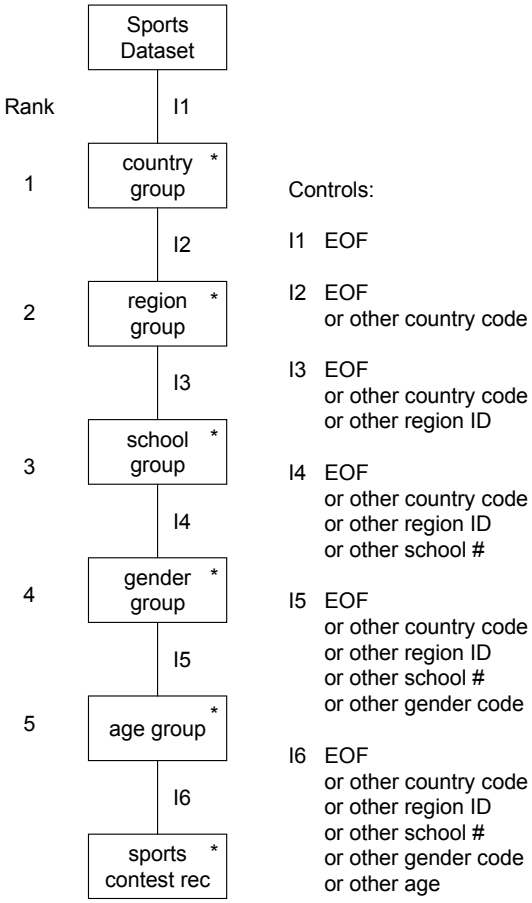


Abbildung 9.1: *Datenstruktur der Sportwettbewerbs-Eingabedaten*

Um dieses Struktogramm zu verstehen, kann man es von unten nach oben lesen:

- Die elementaren Bestandteile der Datei heissen „sports contest rec(ord)". In jedem solchen Satz sind die Identifikatoren aus den ersten fünf Spalten, die sportlichen Ergebnisse und der Name der Schülerin / des Schülers enthalten. Diese innere Struktur ist aber im Struktogramm nicht dargestellt, weil das für den aktuellen Zweck als nicht erforderlich erscheint.
- Alle „sports contest records" mit gleichen Identifikationsmerkmalen gehören zur selben „age group" im fünften Rang. Das drückt die Kontrolle I6 aus. Wechselt das Alter oder das Geschlecht oder die Schulnummer oder die Region oder das Land von einem Satz zum nächsten Satz, oder ist das Dateiende erreicht, so beendet das diese Gruppe.
- Alle teilnehmenden Schülerinnen einer Schule bilden eine „gender group", die aus deren Altersgruppen besteht. Dasselbe gilt analog für die teilnehmenden Schüler. Wenn nur eine einzige männliche oder weibliche Altersgruppe am Wettbewerb teilnimmt, so fallen die beiden Gruppengrenzen zusammen.
- Die Kontrolle I5 interessiert sich folglich nicht mehr für das jeweilige Alter einer Gruppe, sondern nur noch für den Wechsel M ⇨ F beziehungsweise F ⇨ M, sowie für alle Wechsel in den darüberliegenden Gruppenebenen.

- Eine „school group" besteht demnach aus einer F-Gruppe, die von einer M-Gruppe gefolgt wird, oder nur aus einer dieser beiden Gruppen. Schulen, die nur Mädchen unterrichten, melden nur F-Gruppen-Daten, Schulen für Jungen nur M-Gruppen-Daten. In diesen Fällen treffen die Grenzen der „gender group" und der „school group" zusammen.
- Die Kontrolle I4 prüft, ob Land, Region und Schulnummer noch dieselbe wie beim vorhergehenden Satz sind, oder ob das Dateiende erreicht wurde.
- Alle Schulen einer Region gehören zur selben „region group". Diese endet beim Wechsel zu einer anderen Region, zu einem anderen Land, oder am Dateiende, wie I3 vorgibt.
- Sämtliche Regionen eines Landes werden zu einer „country group" zusammengefasst. Wechselt der Inhalt der Spalte „country", so beginnt eine neue „country group". Mit dem Dateiende endet die aktuelle „country group" ebenfalls. I2 prüft diese Bedingungen.
- Das Dateiende begrenzt alle Gruppenebenen. Mit dem Nachlesen, das der Verarbeitung des letzten Satzes folgt, treten alle Gruppenwechselbedingungen einschliesslich der Kontrolle I1 gleichzeitig ein.

Diese Beschreibung ist davon unabhängig, ob die Sortierung nach den einzelnen Identifikationsmerkmalen einheitlich – meist aufsteigend – stattfindet, oder ob mal aufsteigend, mal absteigend sortiert wird. Wichtig ist lediglich, dass die Sortierung überhaupt vorgenommen wird. Die Summen-, Durchschnitts-, Minimal- und Maximalwerte, welche eine Auswertung liefern kann, haben bei unsortierter Eingabe keinerlei brauchbaren Sinn.

An dieser Stelle unterscheidet sich die Gruppenverarbeitung deutlich von der später zu beschreibenden Mischverarbeitung, bei der eine einheitliche – für alle Schlüsselbestandteile nur aufsteigende oder nur absteigende – Sortierung vor allem dann zwingend erforderlich ist, wenn mit konkatenierten Gruppenschlüsseln gearbeitet wird.

Es darf keine zwei aufeinanderfolgenden Zeilen mit denselben Identifikationsmerkmalen und Namen geben, weil sonst unklar wäre, welcher Schüler beziehungsweise welche Schülerin die hier erfassten sportlichen Ergebnisse erzielt hat. Da sich so etwas in der Praxis nicht vollständig ausschliessen lässt, sollte vor dem eigentlichen Auswertungslauf ein Prüfprogramm die Datenqualität verifizieren, damit solche und andere Fehler rechtzeitig bereinigt werden können. Wir haben aber bereits festgelegt, dass wir uns damit hier nicht befassen werden.

Kumulative Iterationskontrollen

Es springt sofort ins Auge, dass bei den Kontrollen von Rang zu Rang nur ein Kriterium hinzukommt und sich sonst nichts ändert. Das sieht sehr ineffizient aus und ist auch hässlich zu codieren. Da wir bereits anhand des vorhergehenden Beispiels die konkatenierten Gruppenschlüssel kennen- und schätzengelernt haben, bietet es sich an, hier ähnlich vorzugehen. Es wäre aber verfrüht, jetzt schon diese Kontrollen umzusetzen, denn wir wissen ja noch gar nicht, welche Ausgabe das Programm erzeugen soll.

9.1.2 Ausgabe-Format

Gold, Silber und Bronze ehren die besten Teilnehmer an sportlichen Wettkämpfen seit der Antike. Die Auswertung der Eingabedatei „Sports Dataset" und einer weiteren Datei mit Schul-Daten erzeugt daher eine Ergebnisliste in Dateiform – zwecks späterer Aufbereitung und Ausgabe – mit folgenden Inhalten:

- Daten der Schule (Name, Adresse ohne Straßenangabe, E-Mail-Adresse), Leistungsdurchschnitt, Anzahlen der Altersgruppen und Teilnehmer, nach F und M getrennt, Medaillenrang und Ergebnisse der besten Sportler(innen) pro Schule
  - Es werden die drei höchsten Punktsummen ermittelt.
  - Bei Gleichstand erhalten jeweils mehr als eine Schülerin / ein Schüler eine bestimmte Medaille.
  - Diese sind dann nicht in einer definierten Folge aufzulisten.
  - Mädchenschulen melden nur F-Gruppen, Schulen für Jungen nur M-Gruppen.
- Rangliste der Schulen mit den besten drei *Durchschnitts*werten pro Land (nur Schul- und Statistik-Daten ohne E-Mail-Adresse)
  - Es müssen mindestens vier Schulen aus dem jeweiligen Land teilgenommen haben.
  - Bei Gleichstand werden alle davon betroffenen Schulen prämiiert.
  - Durchschnitts-Ergebniswerte werden nach der zweiten Nachkommastelle abgeschnitten.
- Rangliste der Schulen mit den besten drei *Durchschnitts*werten insgesamt (analog)
  - Hier gilt bei Gleichstand dasselbe Prinzip, obwohl dieses Ereignis sehr unwahrscheinlich ist.

Das Layout dieser Ausgabedatei ist wie folgt festgelegt:

```
DEUBW0047081101 DEUBW00470811 Hölderlin-Gymnasium
DEUBW0047081102 88214 Ravensburg, Baden-Württemberg, Deutschland
DEUBW0047081103 sekretariat@hoelderlin.rv.bw.schule.de
DEUBW0047081104 Ø 33,50 F 4/129 M 7/432
DEUBW0047081105GM 17 18 16 15 49 Zeisler, Georg
DEUBW0047081106SF 16 15 18 14 47 Rainig, Yvonne
DEUBW0047081107BM 15 17 15 13 45 Zuck, Guenther
[…]
DEUBW0765432101 DEUBW07654321 Paul-Klee-Realschule
DEUBW0765432102 70565 Stuttgart, Baden-Württemberg, Deutschland
DEUBW0765432103 paul-klee-schule@stuttgart.de
DEUBW0765432104 Ø 28,70 F 5/305 M 6/364
DEUBW0765432105GF 15 18 18 15 51 Nenner, Ingrid
DEUBW0765432106SM 16 17 14 17 48 Kawasaki, Jakob
DEUBW0765432107BM 18 14 16 17 47 Biermann, Tobias
[…]
DEUNW0001705001 DEUNW00017050 Schiller-Gesamtschule
DEUNW0001705002 48147 Münster, Nordrhein-Westfalen, Deutschland
DEUNW0001705003 schillerschule@muenster.de
DEUNW0001705004 Ø 31,28 F 7/642 M 8/583
DEUNW0001705005GM 16 19 16 18 53 Düsentrieb, Hubert
DEUNW0001705006SM 15 18 19 15 52 Große-Lashecke, Daniel
DEUNW0001705007BF 16 16 18 14 48 Lallmann, Anita
[…]
```

```
DEU  0000000001GDEUHE00223344 Immanuel Kant Gymnasium
DEU  0000000002 60322 Frankfurt am Main, Hessen, Deutschland
DEU  0000000003 Ø 34,32 F 6/451 M 6/562
DEU  0000000004SDEUBY00123456 Ludwig-Thoma-Gymnasium
DEU  0000000005 86899 Landsberg am Lech, Bayern, Deutschland
DEU  0000000006 Ø 33,65 F 4/372 M 6/522
DEU  0000000007BDEUBW00470811 Hölderlin-Gymnasium
DEU  0000000008 88214 Ravensburg, Baden-Württemberg, Deutschland
DEU  0000000009 Ø 33,50 F 4/129 M 7/432
```

[...] etcetera mit ESP, FRA und ITA bis zum Eingabe-Dateiende, gefolgt von:

```
AAA  0000000001GESPAN02785491 Instituto Federico García Lorca
AAA  0000000002 18000 Granada, Andalucia, España
AAA  0000000003 Ø 36,74 F 3/273 M 3/481
AAA  0000000004SFRA5300112765 Licée Paul Signac
AAA  0000000005 35800 Saint-Briac-sur-Mer, Bretagne, France
AAA  0000000006 Ø 36,65 F 5/284 M 7/462
AAA  0000000007BITALI83007852 Liceo Scientifico Galileo Galilei
AAA  0000000008 18100 Imperia, Liguria, Italia
AAA  0000000009 Ø 35,75 F 3/147 M 5/321
```

Dieses – auf den ersten Blick sonderbar aussehende – Layout ist für weiterverarbeitbare Listenausgaben typisch. Es wäre ja extrem umständlich, wie vor Jahrzehnten eine lange Liste mit Ergebnisdaten auf Endlospapier auszudrucken, die anschliessend von Hand getrennt werden müsste, und die einzelnen Zettel per Post zu versenden. Solche Ausgabedateien mit Gruppenstruktur und Satzzählern können übrigens auch für das Wiederaufsetzen nach einem Lauf-Abbruch sehr nützlich sein.

Nur aus Gründen der Lesbarkeit wurden Leerzeichen und „Bedeutungszeichen" (Ø, F, M) eingefügt.

Jede Schule erhält ihre eigenen Ergebnisse mit den Namen der prämierten Schüler(innen), sowie die Ergebnisse der nationalen und internationalen Durchschnitts-Sieger per E-Mail zugesandt. Dafür muss die Ergebnisdatei zunächst anhand der ersten 15 Stellen jedes Satzes aufsteigend sortiert werden. Die Sort-Ausgabe sieht wie folgt aus:

```
AAA  0000000001GESPAN02785491 Instituto Federico García Lorca
AAA  0000000002 18000 Granada, Andalucia, España
AAA  0000000003 Ø 36,74 F 3/273 M 3/481
AAA  0000000004SFRA5300112765 Licée Paul Signac
AAA  0000000005 35800 Saint-Briac-sur-Mer, Bretagne, France
AAA  0000000006 Ø 36,65 F 5/284 M 7/462
AAA  0000000007BITALI83007852 Liceo Scientifico Galileo Galilei
AAA  0000000008 18100 Imperia, Liguria, Italia
AAA  0000000009 Ø 35,75 F 3/147 M 5/321
DEU  0000000001GDEUHE00223344 Immanuel Kant Gymnasium
DEU  0000000002 60322 Frankfurt am Main, Hessen, Deutschland
DEU  0000000003 Ø 34,32 F 6/451 M 6/562
DEU  0000000004SDEUBY00123456 Ludwig-Thoma-Gymnasium
DEU  0000000005 86899 Landsberg am Lech, Bayern, Deutschland
DEU  0000000006 Ø 33,65 F 4/372 M 6/522
DEU  0000000007BDEUBW00470811 Hölderlin-Gymnasium
```

```
DEU  0000000008 88214 Ravensburg, Baden-Württemberg, Deutschland
DEU  0000000009 Ø 33,50 F 4/129 M 7/432
DEUBW0047081101 DEUBW00470811 Hölderlin-Gymnasium
DEUBW0047081102 88214 Ravensburg, Baden-Württemberg, Deutschland
DEUBW0047081103 sekretariat@hoelderlin.rv.bw.schule.de
DEUBW0047081104 Ø 33,50 F 4/129 M 7/432
DEUBW0047081105GM 17 18 16 15 49 Zeisler, Georg
DEUBW0047081106SF 16 15 18 14 47 Rainig, Yvonne
DEUBW0047081107BM 15 17 15 13 45 Zuck, Guenther
[…]
```

Nun sollte klar sein, warum die Sätze für die internationalen Ergebnisse, die Landesergebnisse und die Einzelergebnisse den jeweiligen 15-stelligen Präfix aus Land-Code, Regions-ID, Schulnummer und laufender Zeilennummer besitzen. In der Spalte 16 stehen die Codes für Gold, Silber und Bronze. In den Schul-Namens-Zeilen wird die Schul-Identifikation wiederholt, damit diese auch nach dem geplanten Abschneiden der 15 Präfix-Zeichen (siehe unten) weiterhin verfügbar ist.

Nach den DEU-Ergebnissen folgen – beim vorliegenden Datenmaterial – die ESP-Gesamtergebnisse, dann die ESP-Einzelergebnisse und so weiter, was durch die beiden Leerzeichen anstelle der Regions-IDs am Anfang der Landes-Gesamtergebnis-Sätze bewirkt wird. Da es kein Land mit dem Code AAA gibt, stehen die internationalen Gesamtergebnisse immer am Anfang der Sort-Ausgabedatei.

Eine Alternative wäre übrigens, die internationalen Ergebnisse einerseits und die Landes-Gesamtergebnisse andererseits in je eine zusätzliche Datei zu schreiben und diese beiden Dateien später zusammen mit den Einzelergebnissen auszuwerten. Das erspart das Sortieren, erhöht aber das Risiko, dass anschliessend durch einen Fehler im Produktionsablauf die falschen Dateien einander zugeordnet werden. Es kommt ja nicht selten vor, dass nachträglich noch Fehler entdeckt werden, die Laufwiederholungen erzwingen, und dann in der wegen Zeitdrucks zunehmenden Hektik versehentlich aktuelle, gültige mit früheren, falschen Daten zusammengeführt werden. Bei sehr hohem Sortiervolumen oder einer kleinen Maschine wäre eine solche Alternative dennoch durchaus denkbar, was hier aber selbst bei Tausenden von Schulen nicht nötig ist.

Durch Abschneiden der ersten 15 Stellen und Zwischenspeichern der rechts davon stehenden Inhalte der ersten beiden Satzgruppen, ein wenig – sprachspezifischer – Aufbereitung und Umstellen der Reihenfolge bei der Ausgabe kann zum Beispiel folgender E-Mail-Inhalt für `sekretariat@hoelderlin.rv.bw.schule.de` erzeugt werden:

```
Hölderlin-Gymnasium
88214 Ravensburg, Baden-Württemberg, Deutschland
Ihre Schulsportwettbewerbsergebnisse 2022:
          Alter  Lauf  Sprung  Wurf  Summe
Gold    M    17    18      16    15     49  Zeisler, Georg
Silber  F    16    15      18    14     47  Rainig, Yvonne
Bronze  M    15    17      15    13     45  Zuck, Guenther
Punktedurchschnitt: 33,50 Altersgruppen/Schüler F 4/129 M 7/432

Nationale Gewinner:
Gold    Immanuel Kant Gymnasium
        60322 Frankfurt am Main, Hessen, Deutschland
        Durchschnitt: 34,32 Altersgr./Schüler F 6/451 M 6/562
Silber  Ludwig-Thoma-Gymnasium
```

```
        86899 Landsberg am Lech, Bayern, Deutschland
        Durchschnitt: 33,65 Altersgr./Schüler F 4/372 M 6/522
************************* Glückwunsch! ************************
Bronze  Hölderlin-Gymnasium
        88214 Ravensburg, Baden-Württemberg, Deutschland
        Durchschnitt: 33,50 Altersgr./Schüler F 4/129 M 7/432
***************************************************************

Internationale Gewinner:
Gold    Instituto Federico García Lorca
        18000 Granada, Andalucia, España
        Durchschnitt: 36,74 Altersgr./Schüler F 3/273 M 3/481
Silber  Licée Paul Signac
        35800 Saint-Briac, Bretagne, France
        Durchschnitt: 36,65 Altersgr./Schüler F 5/284 M 7/462
Bronze  Liceo Scientifico Galileo Galilei
        18100 Imperia, Liguria, Italia
        Durchschnitt: 35,75 Altersgr./Schüler F 3/147 M 5/321
```

Die Jahresangabe in der dritten Zeile stammt aus einem Parameter. Diese Verarbeitung wird ab dem Kapitel „17.2 Datenstrukturen und Ableiten der Programmstruktur“ auf Seite 486 ff modelliert und implementiert.

9.1.3 Datenflussdiagramm

Die „Abbildung 9.2: Datenflussdiagramm der vollständigen Anwendung“ auf Seite 201 zeigt, welchen Weg die Eingabe- und Ergebnisdaten nehmen:

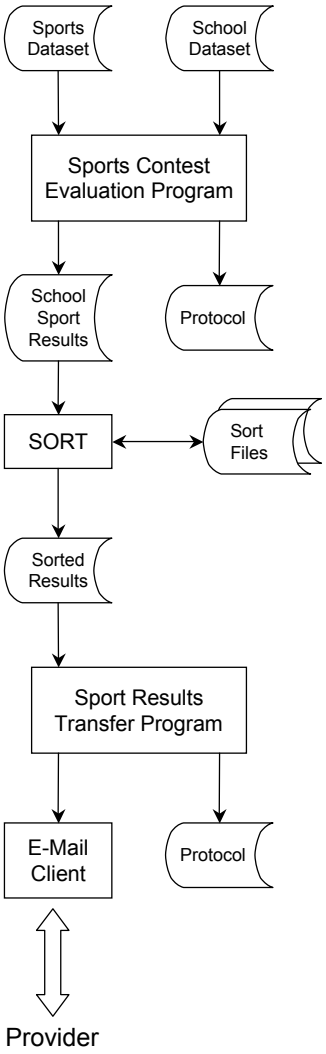


Abbildung 9.2: *Datenflussdiagramm der vollständigen Anwendung*

Die Ergebnisdatei des Auswertungsprogramms „Sports Contest Evaluation Program" heisst „School Sport Results". Sie muss sortiert werden, bevor der E-Mail-Versand der Ergebnisse durch das „Sport Results Transfer Program" stattfinden kann.

9.1.4 Vorläufige Ausgabe-Datenstruktur

Nun wissen wir genug, um die vorläufige Struktur der Ausgabedatei School Sport Results beschreiben zu können (siehe nächste Seite).

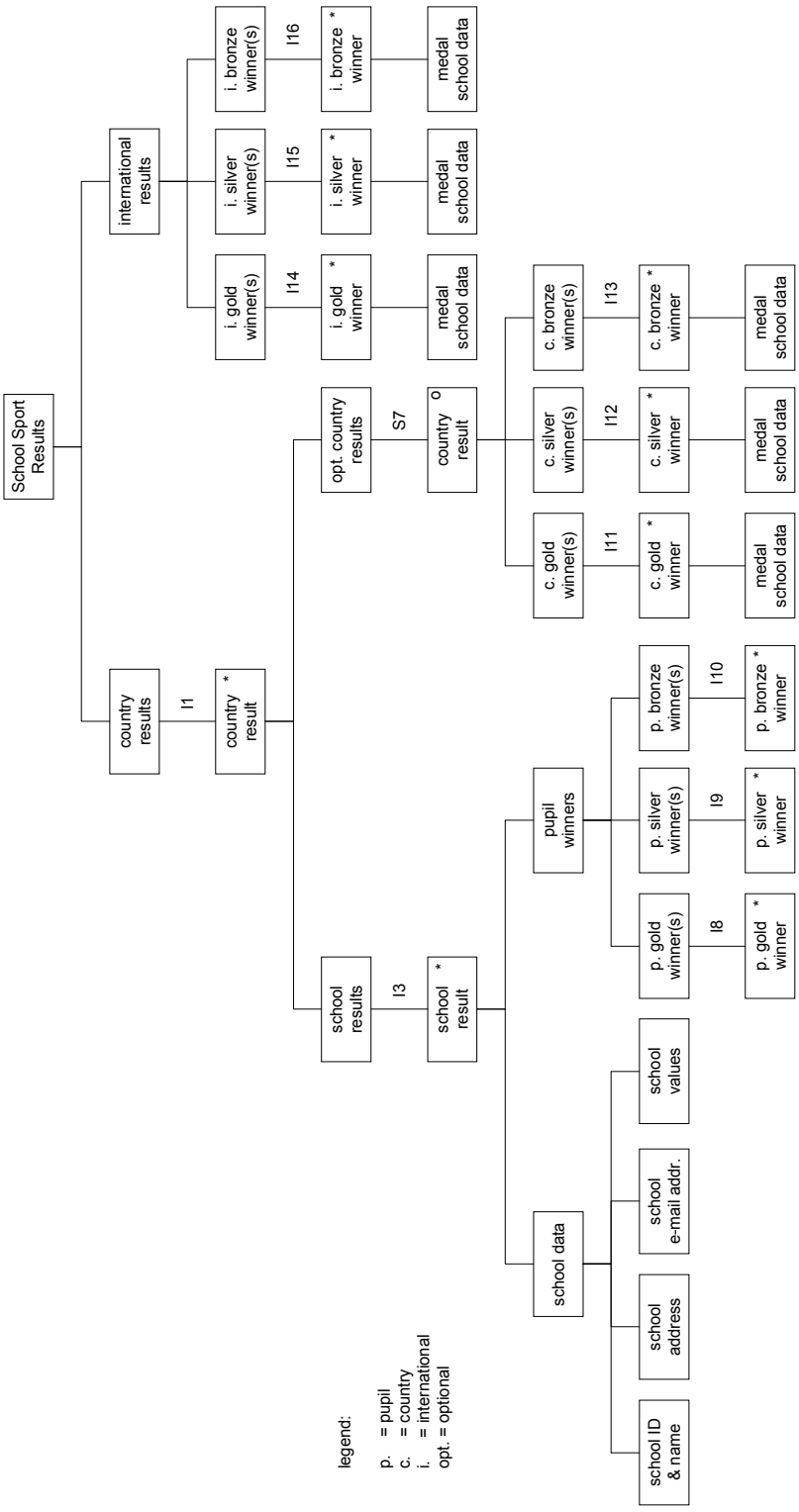


Abbildung 9.3: *Vorläufige Struktur der School Sport Results*

Die neuen Kontrollen lauten:

S7	4 oder mehr teilnehmende Schulen im Land
I8	Ende G-Liste Schule
I9	Ende S-Liste Schule
I10	Ende B-Liste Schule
I11	Ende G-Liste Land
I12	Ende S-Liste Land
I13	Ende B-Liste Land
I14	Ende G-Liste International
I15	Ende S-Liste International
I16	Ende B-Liste International

(G = Gold, S = Silber, B = Bronze)

Beim Betrachten dieses Struktogramms fällt die dreimal auftretende Struktur der Gewinnerlisten sofort auf. Es liegt nahe, hier nach einer Generalisierung zu suchen. Allerdings sind die Ausgaben für die einzelnen Schüler völlig anders als die für die Schulen aufgebaut. Um das Diagramm nicht zu überladen, sind die Details etwas vereinfacht dargestellt. So fehlt beispielsweise die Zeilennummerierung innerhalb der Gruppen mit gleicher Schul-Identifikation (Land-Code, Regions-ID, Schulnummer). Für den Moment ist das Diagramm allerdings völlig ausreichend.

Wir sind jedoch noch weit davon entfernt, die vollständige Programmstruktur abzuleiten. Das liegt daran, dass uns erstens Daten fehlen, denn die Eingabedatei „Sports Dataset“ enthält nur Schulschlüssel, keine Namen und Adressen. Zweitens ist derzeit nicht ersichtlich, wie aus den Einzelergebnissen die Gewinner unter den Schülern / Schülerinnen ermittelt werden können. Drittens ist noch unbekannt, wie das analog für die Durchschnittsergebnisse der Schulen national und international funktionieren soll. Ohne Angaben zu den Schulen und ohne ein Verfahren, mit dem die Gewinner festgestellt werden, können wir nur folgenden Anteil der geplanten Ausgabedatei produzieren:

```
DEUBW0047081104 Ø 33,50 F 4/129 M 7/432
[…]
DEUBW0765432104 Ø 28,70 F 5/305 M 6/364
[…]
DEUBY0012345604 Ø 33,65 F 4/372 M 6/522
[…]
DEUHE0022334404 Ø 34,32 F 6/451 M 6/562
[…]
DEUNW0001705004 Ø 31,28 F 7/642 M 8/583
[…]
ESPAN0278549104 Ø 36,74 F 3/273 M 3/481
[…]
FRA530011276504 Ø 36,65 F 5/284 M 7/462
[…]
ITALI8300785204 Ø 35,75 F 3/147 M 5/321
```

Diese Beispielsätze enthalten nur Daten, die in den Eingabesätzen enthalten sind oder daraus leicht ermittelt werden können. Immerhin basiert darauf auch die Weiterverarbeitung der Schul-Durchschnittsergebnisse, deren Eingabedaten hier aufgelistet sind. Daher erscheint es als sinnvoll, die offensichtlich etwas komplexe

Aufgabe schrittweise zu lösen, indem zunächst gezeigt wird, wie die Eingabedaten in diese reduzierte Ausgabe transformiert werden können. Damit werden bereits wesentliche Elemente der Ranggruppenverarbeitung aus der Theorie in die Praxis umgesetzt. Danach kommen die noch fehlenden Teile der Ausgabedaten dran.

9.2 Erzeugen der reduzierten Ausgabe

9.2.1 Finales Sports Dataset-Struktogramm

Eine wichtige Vereinfachung des ursprünglichen Eingabe-Struktogramms ergibt sich daraus, dass keine Schulergebnisse auf der Regionsebene erzeugt werden sollen. Da die Schulschlüssel landesintern sowieso nur innerhalb ihrer jeweiligen Region als eindeutig angenommen werden dürfen, ist es uns erlaubt, die Ränge 2 und 3 miteinander zu verschmelzen: Aus „region group“ und „school group“ wird die „r-school group“. Um für die weiteren Schritte keine Konfusion zu verursachen, wird parallel dazu die Rangnummerierung angepasst.

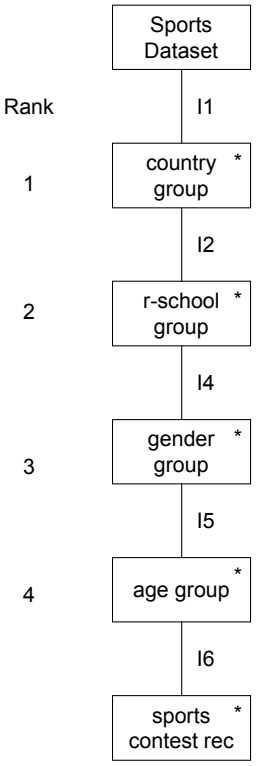
Aus dem Ausgabe-Layout und den dazu formulierten Vorgaben ergeben sich keine weiteren Anforderungen an die Ergebnis-Eingabedatei als diejenigen, die bereits definiert sind. Die momentan nicht herstellbaren Ausgabeelemente – Schulnamen und -adressen, Ranglisten – erfordern zusätzliche Eingabedaten aus einer weiteren Quelle, sowie spezielle Verarbeitungen der „sports contest records“, erzwingen aber keine Modifikation der Sports Dataset-Datenstruktur.

Somit kann schon das finale Eingabe-Struktogramm gezeichnet werden, dem die – kaum überraschende – Struktur der „sports contest records“ beigefügt ist (siehe „Abbildung 9.4: Finales Eingabe-Struktogramm“ auf Seite 205). Die Kontrolle I3 entfällt wegen der zuvor erläuterten Verschmelzung.

Die zugehörige Satzstruktur SpoConD = „Sports Contest Data“ wird aus dem jeweiligen „sports contest rec(ord)“ gefüllt und ist – in Anlehnung an die Syntax von PL/SQL® – wie folgt definiert:

```
SpoConD struct(
    country_code      char(3);
    region_ID         char(2);
    school_number     char(8);
    gender_code       char(1);
    age               num(2);
    sprint_points     num(2);
    jump_points       num(2);
    throw_points      num(2);
    name              char(47);
);
```

Da im Sports Dataset keine Namenslänge mitgeliefert wird, kann diese gegebenenfalls durch Abschneiden der schleppenden Leerzeichen bestimmt werden. Die Begrenzung auf 47 Zeichen ergibt sich aus der maximalen Ergebniszeilen-Länge von 80 Zeichen, die für einen – später folgenden – Ausflug in die Vergangenheit nötig ist.



Sports contest record format:

```
CCCRRSSSSSSSS G AG RN LJ LT NAME (length <= 47)
|   |   |      | |  |  |  |
|   |   |      | |  |  |  LT = long throw
|   |   |      | |  |  LJ = long jump
|   |   |      | |  RN = 100m / 50m sprint
|   |   |      | AG = age
|   |   |      G = gender code
|   |   SSSSSSSS = school number  )
|   RR = region ID                } school ID
CCC = country code                )
```

Abbildung 9.4: *Finales Eingabe-Struktogramm*

Die Korrespondenzen zwischen dem finalen Eingabe-Struktogramm und dem – stark geschrumpften, vorläufigen – Ausgabe-Struktogramm sehen wie folgt aus:

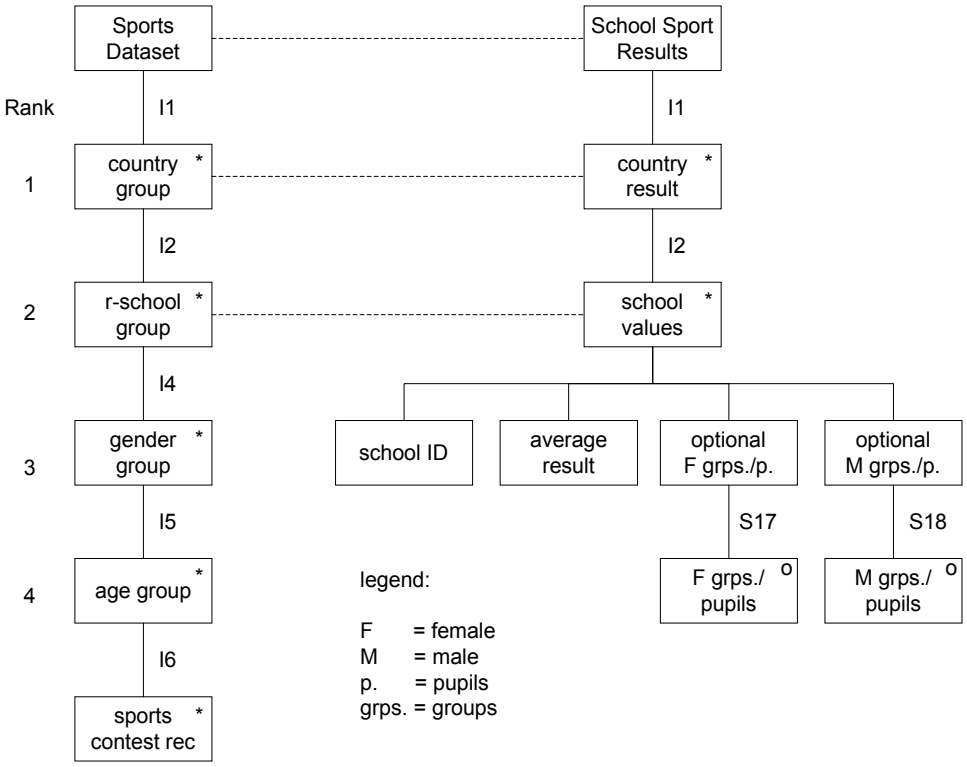


Abbildung 9.5: *Vorläufige Korrespondenzen zwischen den Sportwettbewerbs-Beständen*

Die neuen Kontrollen lauten:

S17 F-Age-Groups > 0
S18 M-Age-Groups > 0

Die Beiträge zu den Ausgabeelementen rechts neben der School ID werden durch geeignete Operationen ermittelt. Die Zeilennummer ist stets 04.

9.2.2 Vereinfachte Programmstruktur ohne Operationen

Durch die Verschmelzung der korrespondierenden Strukturblöcke entsteht die vereinfachte Programmstruktur in der „Abbildung 9.6: Vorläufige Programmstruktur“ auf Seite 207.

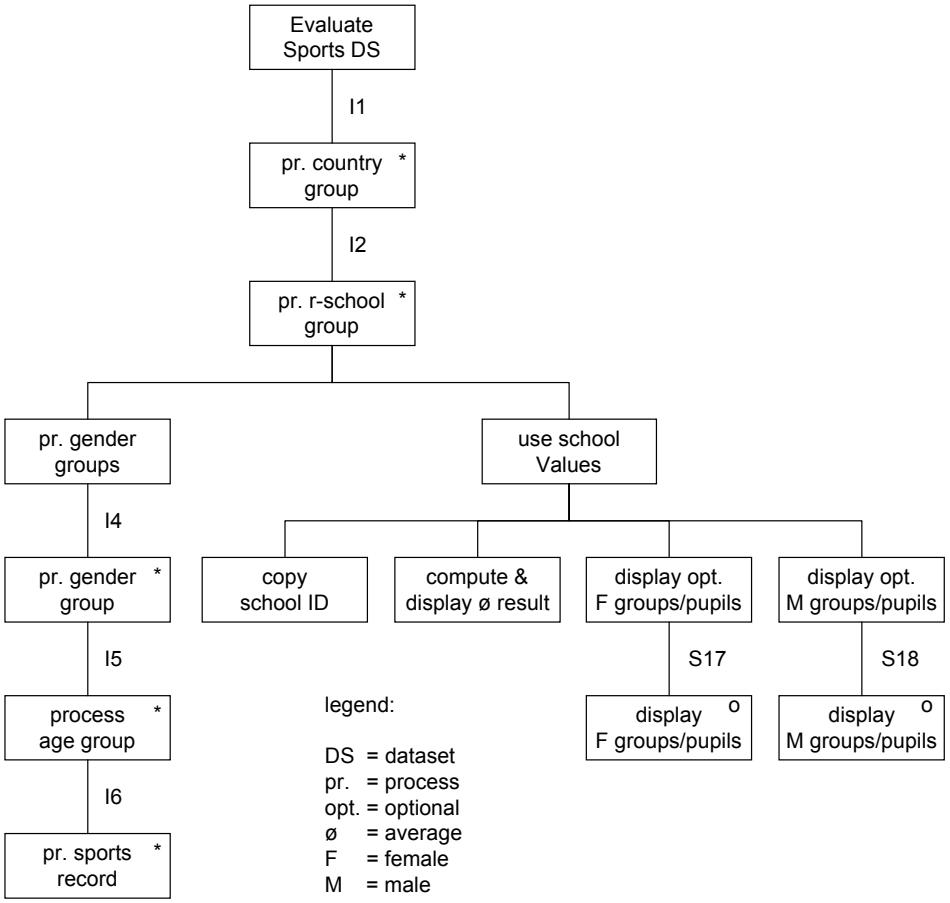


Abbildung 9.6: *Vorläufige Programmstruktur*

9.2.3 Umwandlung der Iterationskontrollen I2 bis I6

Bevor die Operationen definiert werden können, empfiehlt es sich, die kumulativen Iterationskontrollen umzuformulieren. In der bisherigen Form sind sie sehr unhandlich, umständlich zu codieren, und verbrauchen unnötige Ausführungszeit, weil die mit „oder" verknüpften Einzelbedingungen sämtlich geprüft werden müssen, bis eine von ihnen den Wert `true` ergibt. Relativ viel Code befasst sich mit relativ wenigen Bytes. Daher liegt es nahe, für alle Rangstufen je einen konkatenierten Schlüssel zu definieren und Änderungen dieser Schlüssel abzufragen.

Die folgende Tabelle zeigt die Zusammensetzung dieser Schlüssel und Beispiele dazu:

	EOF Indicator	Country Code	Region ID	School #	Gender Code	Age	Example
rank_1_key	X	X					`DESP`
rank_1_2_key	X	X	X	X			`DESPAN02785491`
rank_1_3_key	X	X	X	X	X		`DESPAN02785491F`
rank_1_4_key	X	X	X	X	X	X	`DESPAN02785491F17`

Von jedem dieser Schlüssel wird ein aktuelles und ein altes Exemplar benötigt, um die Vergleiche ausführen zu können. Die Kontrollen in der obigen Programmstruktur lauten daher wie folgt:

I1 EOF
I2 new_rank_1_key > old_rank_1_key
I4 new_rank_1_2_key > old_rank_1_2_key
I5 new_rank_1_3_key > old_rank_1_3_key
I6 new_rank_1_4_key > old_rank_1_4_key
S17 F-Age-Groups > 0
S18 M-Age-Groups > 0

9.2.4 Definition und Zuordnung von Operationen

Die Liste der Operationen kann ebenfalls begonnen werden:

Op.	Bedeutung	Kommentar
1	Sports-Dataset öffnen	txt-Datei wird erwartet
2	Results-Dataset öffnen	
3	Sports-Record lesen, konkatenierte Rangschlüssel zuweisen	Einzelfelder extrahieren, gegebenenfalls Namenslänge bestimmen
4	Result-Satz 04 (RR04) ausgeben	Satz mit verdichteten Ergebnissen
5	Sports-Dataset schließen	
6	Results-Dataset schließen	
7	Stopp	
8	old_rank_1_key := new_rank_1_key	Bisherigen Rang-1-Gruppenschlüssel sichern
9	old_rank_1_2_key := new_rank_1_2_key	Bisherigen Rang-1-2-Gruppenschlüssel sichern
10	old_rank_1_3_key := new_rank_1_3_key	Bisherigen Rang-1-3-Gruppenschlüssel sichern
11	old_rank_1_4_key := new_rank_1_4_key	Bisherigen Rang-1-4-Gruppenschlüssel sichern
12	countries := 0	Statistik-Zähler (auf Vorrat definiert)
13	countries := countries + 1	
14	schools := 0	Schulen im Land
15	schools := schools + 1	
16	F_age_groups := 0	
17	M_age_groups := 0	
18	F_pupils := 0	
19	M_pupils := 0	
20	pupils := 0	Schüler in einer „age group“
21	pupils := pupils + 1	

Op.	Bedeutung	Kommentar
22	**if** gender_code = `'F'` **then** F_age_groups := F_age_groups + 1; F_pupils := F_pupils + pupils **else** M_age_groups := M_age_groups + 1; M_pupils := M_pupils + pupils **fi**	gender_code = 15. Stelle von old_rank_1_4_key (Index: [14])
23	point_sum := SpoConD.sprint_points + SpoConD.jump_points + SpoConD.throw_points	
24	school_points := 0	
25	school_points := school_points + point_sum	
26	RR04 := Substr(old_rank_1_2_key, 1, 13) + `'04'`	school ID + Zeilennummer 04 (Start-Index von old_rank_1_2_key = 0)
27	avg_school_points := school_points / (F_pupils + M_pupils)	Ergebnis nach 2 Nachkommastellen abschneiden
28	RR04 := RR04 + `' Ø '` + String(avg_school_points)	Anhängen: `' Ø nn,nn'` (mit Umwandlung Zahl ⇨ String)
29	RR04 := RR04 + `' F '` + String(F_age_groups) + `'/'` + String(F_pupils)	Anhängen: `' F nn/nnnn'` (mit variablen Stellenzahlen)
30	RR04 := RR04 + `' M '` + String(M_age_groups) + `'/'` + String(M_pupils)	Anhängen: `' M nn/nnnn'` (mit variablen Stellenzahlen)

Die „Abbildung 9.8: Durchlauf bis nach dem Einlesen des zweiten Satzes“ auf Seite 211 zeigt die mit diesen Operationen bestückte Programmstruktur.

9.2.5 Durchlaufanalyse anhand der vereinfachten Programmstruktur

Der Ausnahmefall der leeren Eingabedatei lohnt keine nähere Befassung, weil dann nur die Operationen 1,2,3,12, die Kontrolle I1 und die Op. 5,6,7 durchlaufen werden.

Die Durchlaufanalyse beginnt daher mit dem ersten Satz der Daten zum Liceo Federico Fellini. Die Beispiel-Tabelle in „9.1.1 Vorläufige Eingabe-Datenstruktur“ auf Seite 192 enthält am Anfang folgende Daten:

country	region	school #	gender code	age	100m / 50m sprint	long jump	long throw	name
ITA	SI	45008132	F	15	15	16	7	Coglitore, Laura
ITA	SI	45008132	F	15	4	7	6	Peritore, Mariuccia
ITA	SI	45008132	F	16	3	7	6	Di Stefano, Serena
ITA	SI	45008132	F	16	3	9	8	Lo Porto, Mariuccia
ITA	SI	45008132	F	16	18	15	17	Prestìa, Angela
ITA	SI	45008132	F	17	10	8	12	Castagnola, Michela

Diesen Zeilen entsprechen folgende Sätze der Test-Eingabedatei:

```
ITASI45008132 F 15 15 16 07 Coglitore, Laura
ITASI45008132 F 15 04 07 06 Peritore, Mariuccia
ITASI45008132 F 16 03 07 06 Di Stefano, Serena
ITASI45008132 F 16 03 09 08 Lo Porto, Mariuccia
ITASI45008132 F 16 18 15 17 Prestìa, Angela
ITASI45008132 F 17 10 08 12 Castagnola, Michela
```

Das Struktogramm in der „Abbildung 9.8: Durchlauf bis nach dem Einlesen des zweiten Satzes“ auf Seite 211 zeigt die vereinfachte Programmstruktur mit Operationen. Sie ist hinsichtlich der eingabeorientierten Operationen absolut typisch für Ranggruppenverarbeitungen aufgebaut. Da solche Struktogramme für JSP-Neulinge erfahrungsgemäß nicht auf Anhieb verständlich sind, ist bereits der Beginn eines ersten Durchlaufs eingezeichnet. Der Strukturblock „use school values“ wurde aus Platzgründen ausgegliedert:

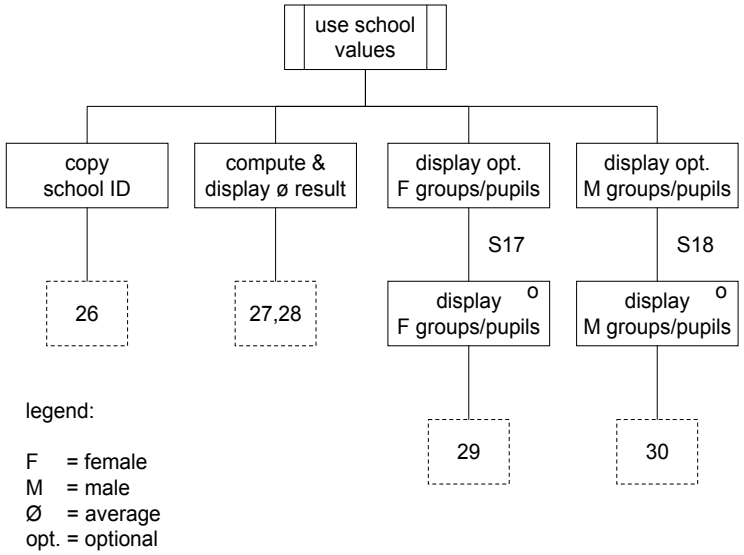


Abbildung 9.7: *Strukturblock „use school values“*

Beginn der Durchlaufanalyse

Die Plazierung der Operationen 8 bis 11 liegt auf der Hand: Am jeweiligen Gruppenanfang, unmittelbar vor der Schleifenkontrolle der Gruppe, setzt eine Zuweisung den neuen und den alten Rangschlüssel gleich. Somit kann keine dieser Kontrollen das Ergebnis `true` abliefern und der Durchlauf führt hinab bis auf die Stufe, auf der die elementaren Satzinhalte verarbeitet werden. Das sieht redundant aus, ist aber innerhalb der JSP-Methode unumgänglich.

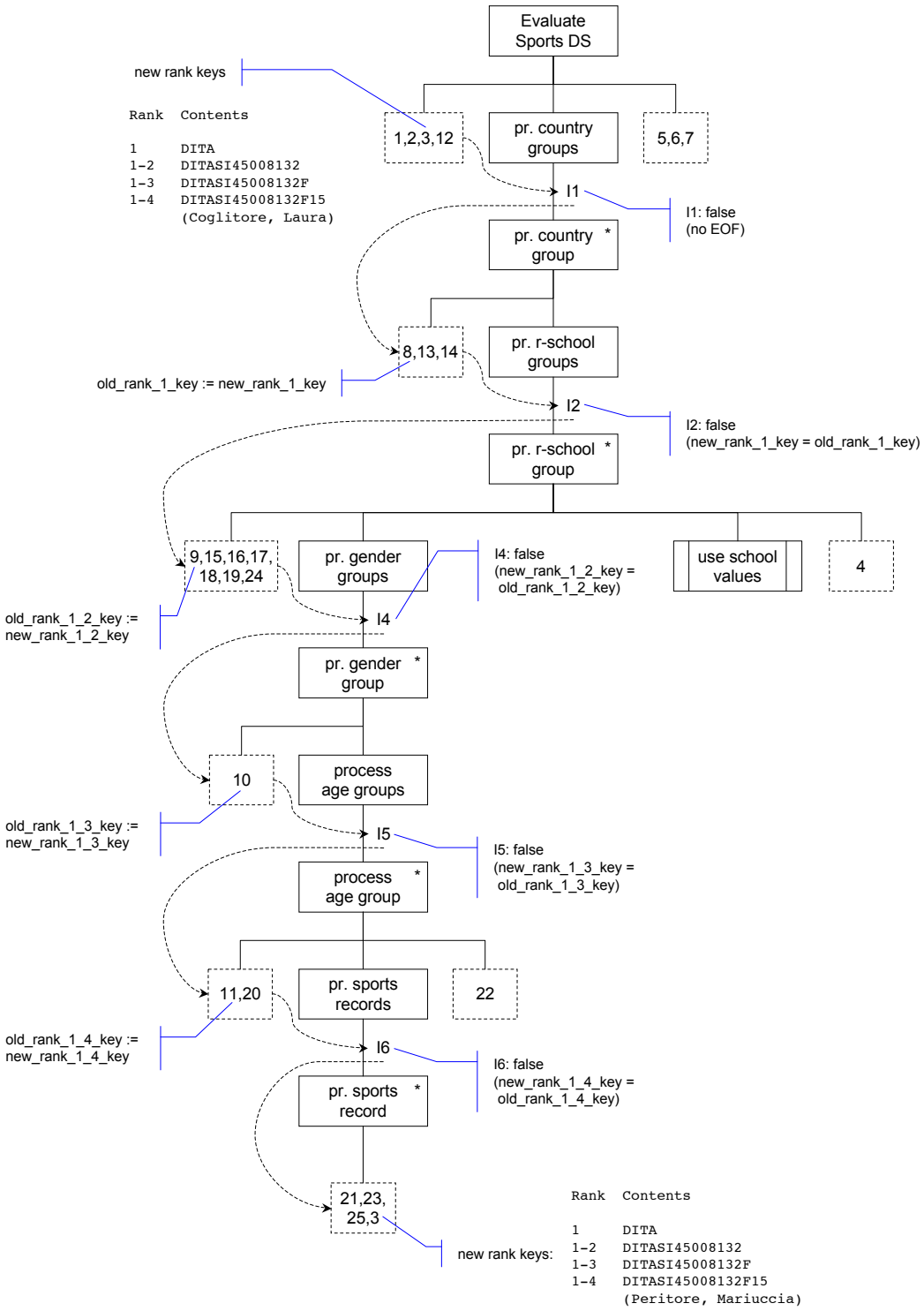


Abbildung 9.8: *Durchlauf bis nach dem Einlesen des zweiten Satzes*

Fortsetzung im selben Rang

Nach diesem Eröffnungszug kommt es darauf an, ob der ganz unten nachgelesene Satz einen anderen Gruppenschlüssel – auf irgendeinem Rang – als der aktuelle Satz aufweist. Das ist im vorliegenden Beispiel nicht der Fall. Daher verharrt die Verarbeitung auf dem untersten Rang. Die folgende Grafik zeigt, wie der zweite Satz – auf dem strichlierten Weg von 1 zu 2 – nach der Kontrolle I6 verarbeitet und dann der dritte Satz – zur Schülerin Di Stefano, Serena – eingelesen wird:

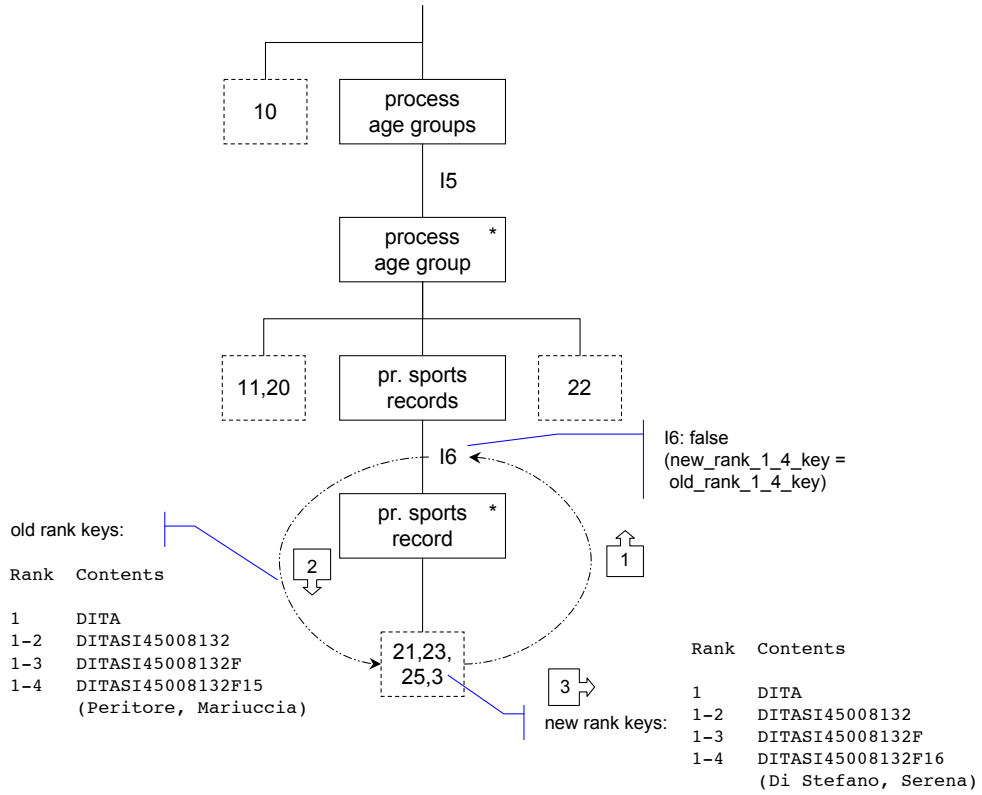


Abbildung 9.9: *Durchlauf bis nach dem Einlesen des dritten Satzes*

Erster Gruppenwechsel: Beendigung der Rang-4-Schleife

Nun liegt ein Gruppenwechsel auf dem untersten Rang (4) vor, da die Altersgruppe 15 beendet ist und die Altersgruppe 16 beginnt. Die Kontrolle I6 ist daher erfüllt. Der folgende Verarbeitungsabschnitt wird in zwei Schritten gezeigt, weil die Grafiken sonst zu verwirrend sind. Zunächst findet der Durchlauf bis zur I5-Kontrolle statt. Das zeigt die „Abbildung 9.10: Durchlauf nach dem Gruppenwechsel bis hoch zu I5“ auf Seite 213.

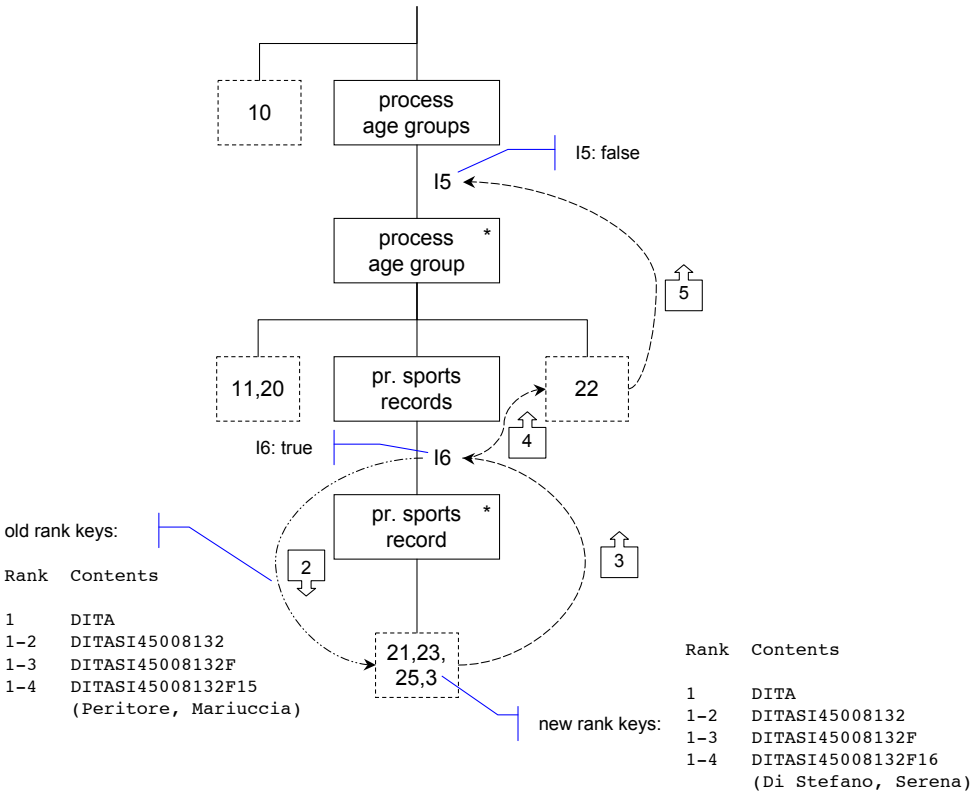


Abbildung 9.10: *Durchlauf nach dem Gruppenwechsel bis hoch zu I5*

Erster Gruppenwechsel: Aufstieg zur Rang-3-Kontrolle

Jetzt kommt erstmals die Ranggruppenlogik wirklich zur Anwendung. Der Durchlauf beginnt mit der Markierung 3 und wird wegen der erfüllten Kontrolle I6 über die Operation 22 mit der Kontrolle I5 fortgesetzt. Deren Endebedingung ist aber nicht erfüllt, weil sich bis auf die Altersgruppe nichts geändert hat.
Somit geht es mit den Operationen 11 und 20 weiter – siehe „Abbildung 9.11: Durchlauf nach dem Gruppenwechsel und Beginn der neuen Altersgruppe“ auf Seite 214.

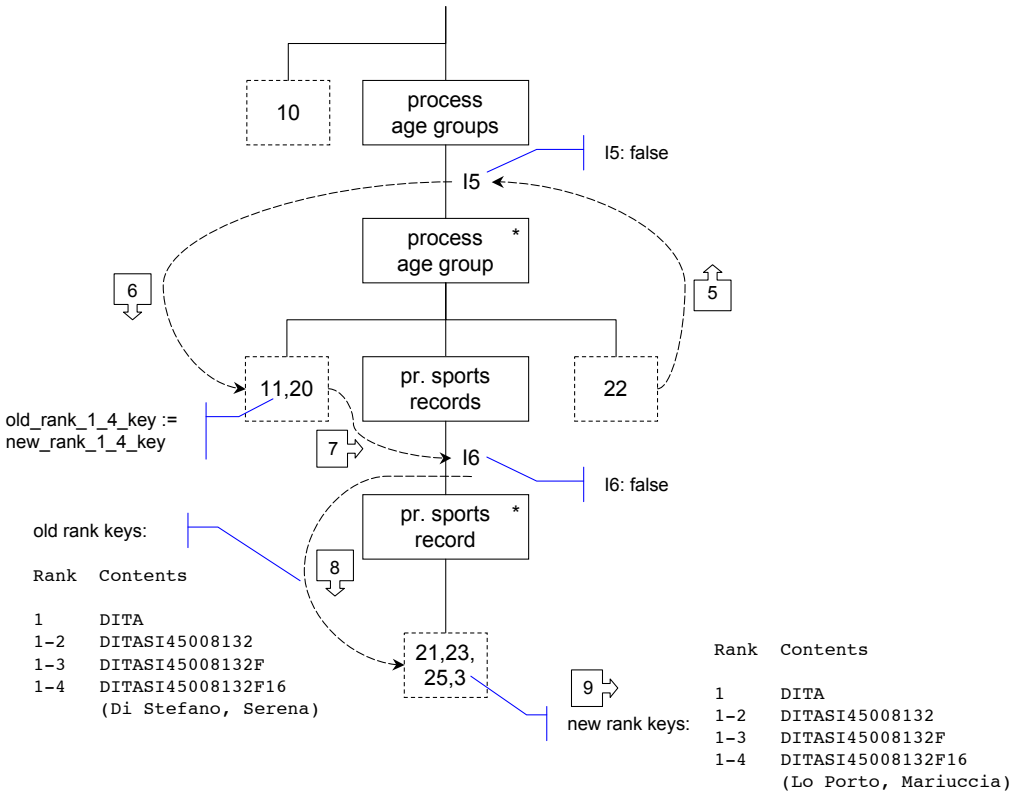


Abbildung 9.11: *Durchlauf nach dem Gruppenwechsel und Beginn der neuen Altersgruppe*

Erste Rückkehr zur Rang-4-Schleife

Die Op. 11 macht wieder die Rangschlüssel identisch. Somit ist die Kontrolle I6 nicht mehr erfüllt und der Satz wird verarbeitet. Das Nachlesen – Markierung 9 – ergibt, dass kein Gruppenwechsel vorliegt. Also ist I6 erneut nicht erfüllt und die Schleife wird fortgesetzt. Dann wird folgender Satz nachgelesen:

```
ITASI45008132 F 16 18 15 17 Prestìa, Angela
```

Auch dieser Satz bewirkt keinen Gruppenwechsel. Erst der Satz

```
ITASI45008132 F 17 10 08 12 Castagnola, Michela
```

verursacht einen erneuten Gruppenwechsel nach dem oben dargestellten Schema. So geht es bis zu folgendem Satz weiter:

```
ITASI45008132 F 18 05 02 04 Tripòdi, Elvira
```

Dieser Satz ist der letzte F-Satz zur Altersgruppe 18. Es folgt der erste M-Satz zur Altersgruppe 13:

```
ITASI45008132 M 13 09 06 09 Ciccione, Pippo
```

Zweiter Gruppenwechsel: Beendigung der Rang-4- und Rang-3-Schleifen

Dies ist ein Gruppenwechsel im Rang 3. Die folgende Grafik zeigt den Durchlauf für diesen Satz bis zur I4-Kontrolle:

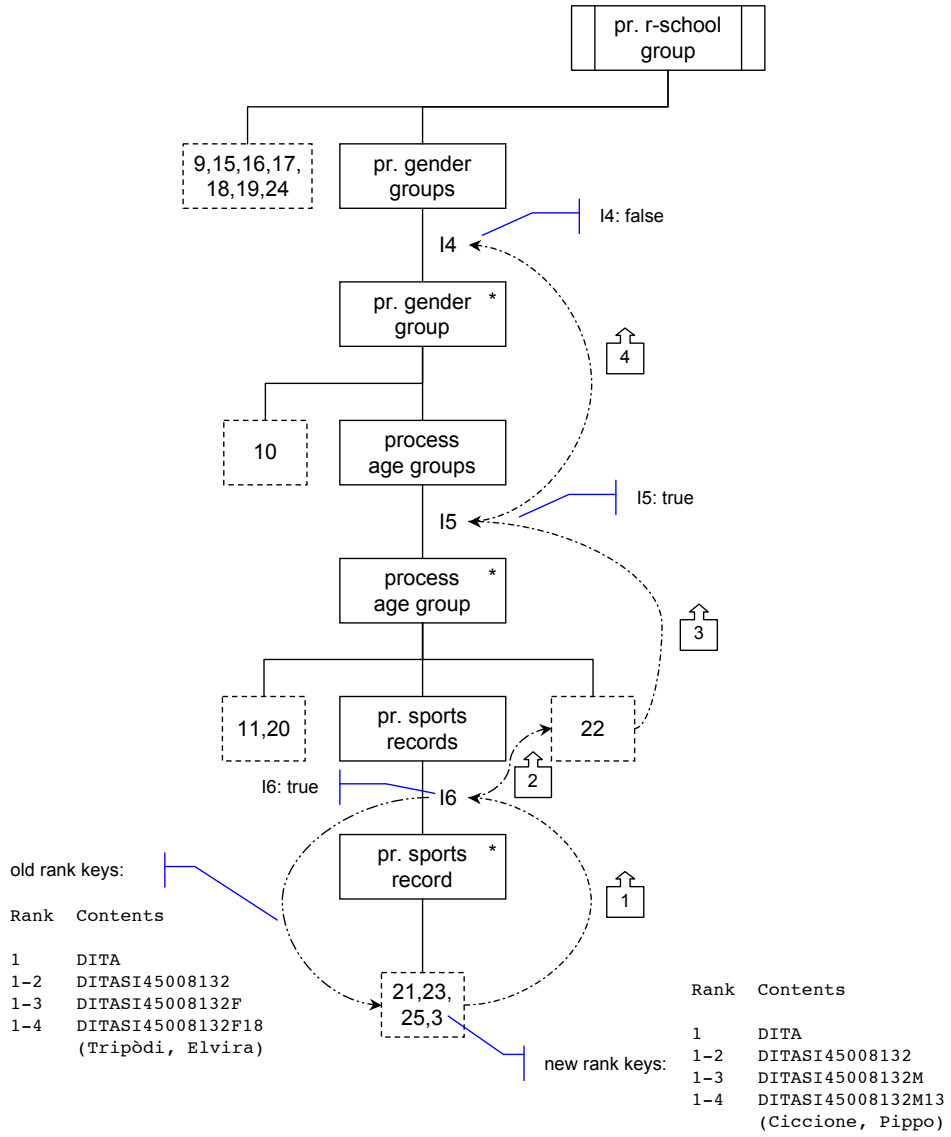


Abbildung 9.12: *Durchlauf nach dem Gruppenwechsel im Rang 3 bis hoch zu I4*

Genau betrachtet, gibt es zugleich einen Gruppenwechsel im vierten Rang (von 18 auf 13). Der Vergleich der konkatenierten Gruppenschlüssel durch I6 erstreckt sich jedoch über alle Ränge, so dass dieser zusätzliche Wechsel keine weitere Bedeutung hat. In einer „handgestrickten" Gruppenverarbeitung könnte man versucht sein, allein aufgrund dieses Zusammentreffens zunächst einen Gruppenwechsel im Rang 4 und dann im Rang 3 zu konstatieren, bei einem Übergang 18 ⇨18 aber nicht. Das wäre aber völlig falsch, da ein Gruppenwechsel auf einem höheren Rang – mit kleinerer Rangnummer – auch alle darin eingeschlossenen Gruppen beendet,

unabhängig davon, ob es dort zufällig weitere Wechsel gibt. Die Altersgruppe 13 in der neuen M-Gruppe hat – datentechnisch – keinerlei Bezug zur Altersgruppe 18 in der beendeten F-Gruppe.

Solche Missverständnisse und Irrtümer vermeidet die (Rang-)Gruppenverarbeitung nach Jackson zuverlässig. Nachdem Sie dieses Kapitel durchgearbeitet haben, werden Sie sich ohne wurmende Zweifel diesem Verfahren anvertrauen und es sorgfältig anwenden, weil es Sie sicher zum Ziel führt.

Zweite Rückkehr zur Rang-4-Schleife

Dieser größere Gruppenwechsel vollzieht sich ganz analog zum Rang-4-Wechsel. Nach der I4-Kontrolle geht es wie folgt weiter:

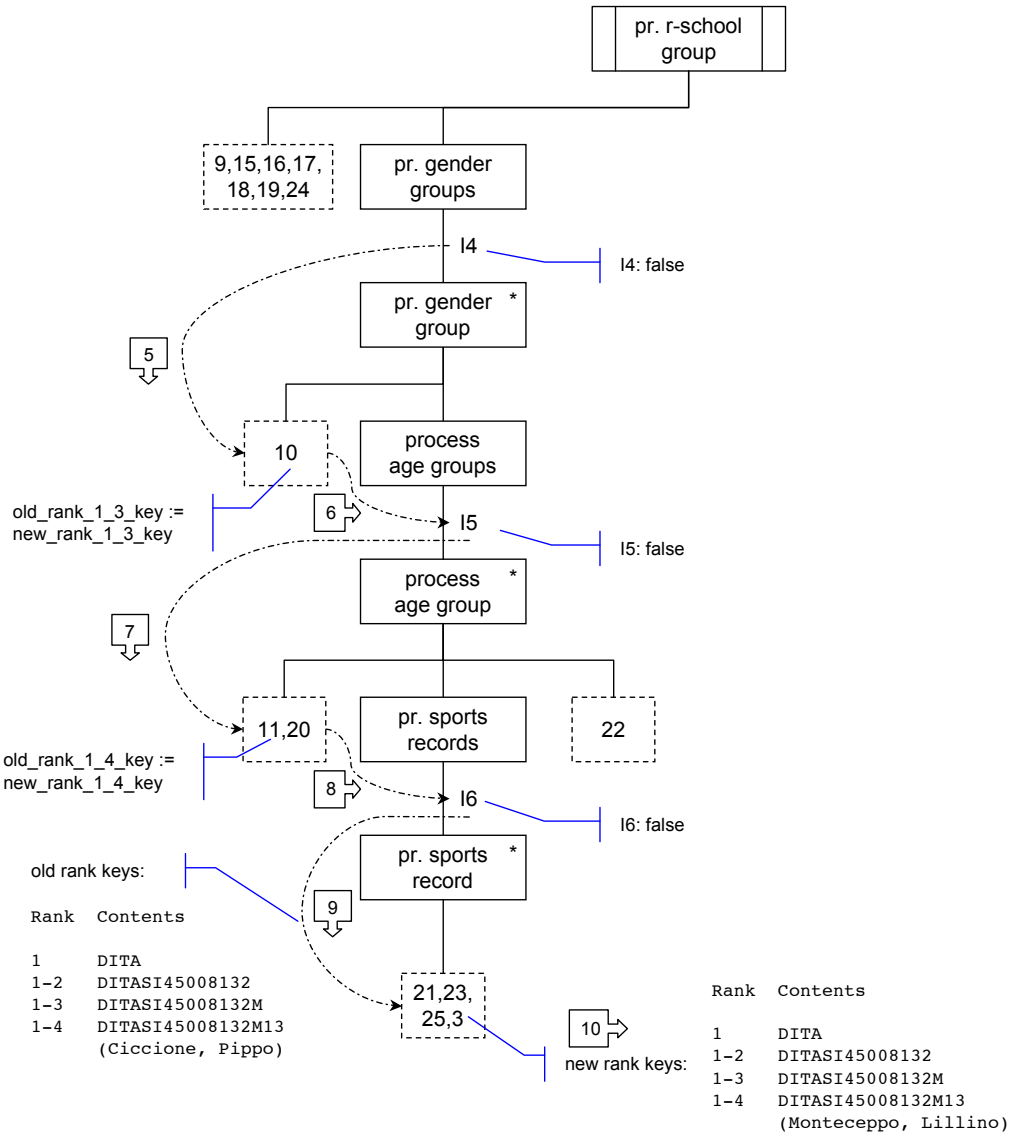


Abbildung 9.13: *Abstieg auf die unterste Ebene in der neuen Gender-Gruppe*

und so weiter und so fort…

Die Verarbeitung aller M-Sätze zur selben Schule erbringt keine neuen Erkenntnisse. Daher sehen wir uns jetzt einen Wechsel im Rang 2 – den Übergang zu einer neuen Schule – an. Damit ist ja die Ausgabe des ersten Ergebnissatzes zur bisherigen Schule verbunden. Bei diesem Wechsel kann sowohl die Region als auch die Schulnummer wechseln. Ändert sich die Region, so ist auch eine neue Schulnummer zu erwarten, aber das ist nicht zwingend.

Der letzte Satz zum Liceo Federico Fellini in 92014 Porto Empedocle, Sicilia, Italia sei:

```
ITASI45008132 M 18 12 11 16 Sutera, Nenè
```

Ihm folgt – als Beispiel – der erste Satz zur neuen Schulgruppe für das Istituto Tecnico Nautico D. Dolfin, 30124 Venezia, Veneto, Italia:

```
ITAVE31005010 M 18 16 13 17 Brunelli, Lilio
```

Das „Abbildung 9.14: Durchlauf nach Wechsel der Schul-Gruppe bis hoch zu I2“ auf Seite 218 zeigt den dadurch angestoßenen Durchlauf.

Dritter Gruppenwechsel: Beendigung der Rang-4-, Rang-3 und Rang-2-Schleifen

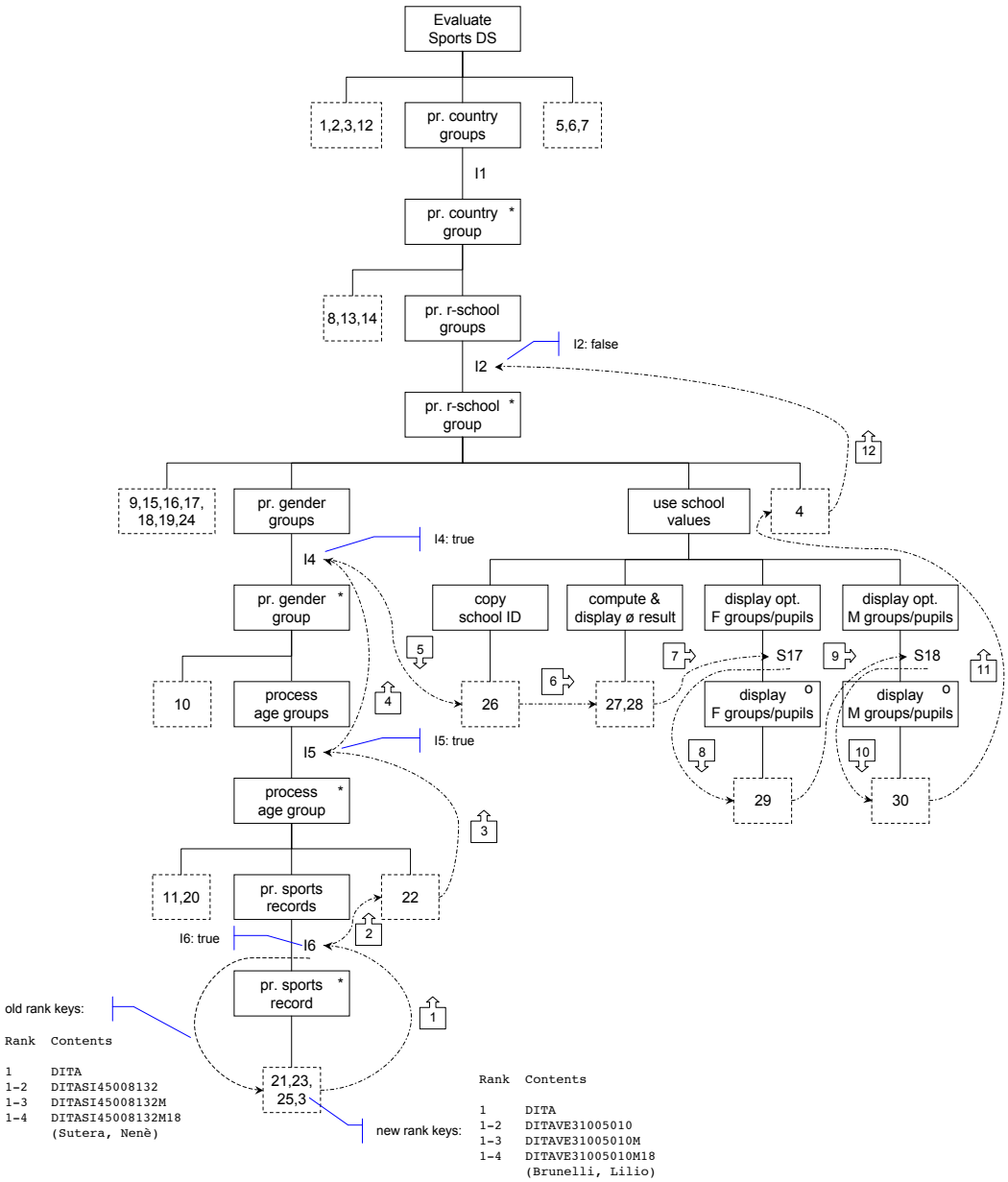


Abbildung 9.14: *Durchlauf nach Wechsel der Schul-Gruppe bis hoch zu I2*

Dieses Diagramm sollte Ihnen auf Anhieb verständlich sein. Die in den Rängen 4 und 3 geübten Prinzipien werden hier auf den Rang 2 angewandt. Zusätzlich findet die Ausgabe des Ergebnissatzes statt, den der alte Rang-1-2-Schlüssel `ITASI45008132` einleitet. Das demonstriert einen weiteren Nutzen der JSP-Gruppenlogik.

Der neu eingelesene Satz dient zunächst nur als Auslöser für die Gruppenwechsel „von unten nach oben", also vom Rang 4 zum Rang 2. Die „Abbildung 9.15: Erneuter Abstieg auf die unterste Ebene mit den Daten zur neuen Schule" auf Seite 219 zeigt, wie er verarbeitet und anschliessend erneut nachgelesen wird.

Dritte Rückkehr zur Rang-4-Schleife

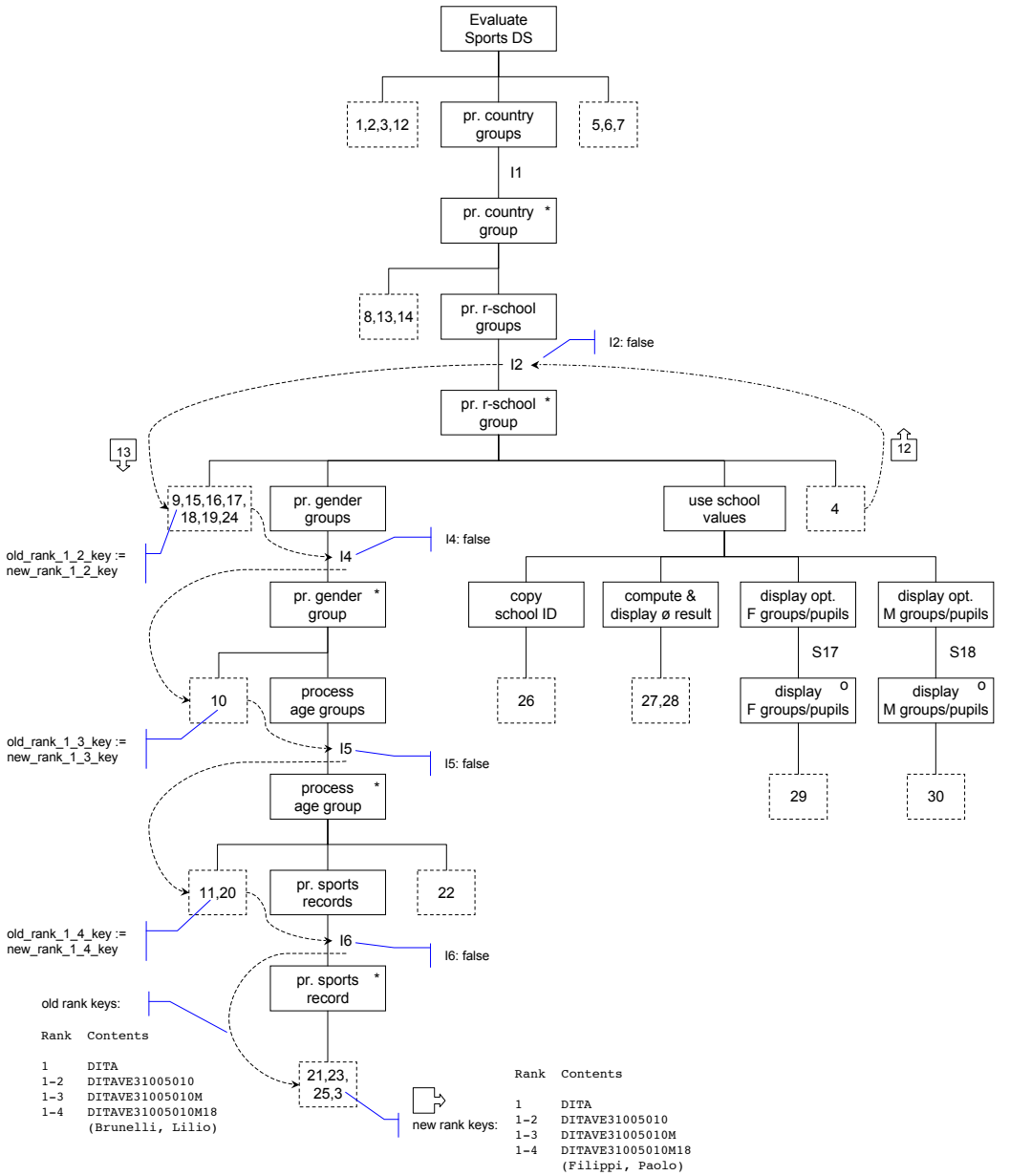


Abbildung 9.15: *Erneuter Abstieg auf die unterste Ebene mit den Daten zur neuen Schule*

Der Durchlauf bei einem Land-Wechsel (Rang 1) verläuft vollständig analog. In diesem Fall ist die Kontrolle I2 erfüllt und es wird die Kontrolle I1 geprüft, bevor die Verarbeitung wieder bis in die I6-Schleife hinabsteigt. Wichtig ist, dass auch in diesem Fall ein Ergebnissatz zum alten Rang1-2-Schlüssel erzeugt wird.

Am Eingabe-Ende ist zusätzlich auch I1 erfüllt. Danach folgen nur noch die Operationen 5,6,7. Es lohnt sich nicht, für diese beiden Ereignisse die Durchlaufdiagramme zu zeichnen.

Stattdessen erscheint es als angebracht, die gezeigten Durchläufe etwas zu verallgemeinern:

- Die hier vorgestellte Ereignisfolge ist beispielspezifisch. In einer gewöhnlichen Ranggruppenverarbeitung können jederzeit Gruppenwechsel auf beliebigen Rängen und in beliebiger Folge stattfinden.
- Vollzieht man solche Fälle grafisch nach, so oszilliert die Verarbeitung über die Rangstufen hinweg auf und ab, verharrt aber auch häufig in der untersten Schleife, wo die Eingabesätze verarbeitet werden.
- Eine Gruppe kann aus einem einzigen Satz bestehen, der beliebige Ranggruppenwechsel auslöst.
- Beim Schleifeneintritt sind die Kontrollen der – wie immer abweisenden – Gruppenschleifen jeweils durch das rechtzeitige Gleichsetzen des alten mit dem neuen Gruppenschlüssel auf `false` geschaltet. Daher wirken sich Gruppenwechsel, die ja wegen der Position der Nachlese-Operation nur in der innersten Schleife auftreten können, bei den anschliessenden Schleifenbeendigungen „von unten nach oben" nur bis zu derjenigen Kontrolle aus, deren Schlüsselabschnitt – vom Anfang her gelesen – unverändert geblieben ist. Danach taucht die Verarbeitung wieder bis in die innerste Schleife hinab, wo die nachgelesenen Daten endlich vollständig verarbeitet werden.
- Das bedeutet, dass bei der aufsteigenden Bewegung – zum Beispiel beim Rang-2-Wechsel entlang der Markierungen 1 bis 12 – zwar schon der neue Satzinhalt im Eingabebereich steht, aber noch Abschlussaktivitäten für die soeben zu Ende gegangene Gruppe stattfinden. Diese greifen auf die Ergebniswerte der abgeschlossenen Gruppe(n) und auf deren Schlüssel zu, um gültige Resultate zu liefern. Inhalte des nachgelesenen Satzes spielen bei dieser Bewegung im Allgemeinen keine Rolle.
- Bei der anschliessenden absteigenden Bewegung nach der ranghöchsten Kontrolle, die `false` ergibt, werden Teile der Inhalte des neuen Satzes verwendet. Meist sind das Abschnitte seines vollen Schlüssels, welche die alten Rangschlüssel überschreiben.
- Nach der Erkennung des Eingabe-Endes findet nur noch die aufsteigende Bewegung statt. Es liegen keine nachgelesenen Daten mehr vor, was bei kunstvollen Konstrukten zu beachten ist, die in dieser Phase schon neue Daten einzubeziehen versuchen. Wenn tatsächlich der Inhalt eines gruppenwechsel-auslösenden Satzes auf die Ergebnisermittlung für die bisherige Gruppe Einfluss nehmen muss, ist eine dementsprechende Datenstruktur zu erstellen, die auch den Fall des – gegebenenfalls unerwarteten – Eingabe-Endes berücksichtigt.
- Mit der JSP-Gruppenlogik ist es verbunden, dass die Schlüsselbestandteile mehrfach kopiert beziehungsweise geprüft werden, und zwar um so öfter, je hochrangiger sie sind. Diese Redundanz mag aus der Performance-Sicht als störend erscheinen, lässt sich aber nicht vermeiden.

9.2.6 Implementierung der vereinfachten Programmstruktur

Die zuverlässige Beherrschung der Gruppenverarbeitung ist zumindest für die kommerzielle Datenverarbeitung von hoher Bedeutung. Praktisch jede Verarbeitung, die mit großen Datenmengen konfrontiert ist, verwendet irgendeine Gruppenlogik oder könnte davon profitieren. Das hier gewählte Beispiel, der Internationale Schulsportwettbewerb, weist sowohl Züge einer allgemeinen Ranggruppenverarbeitung als auch Besonderheiten auf. Daher erscheint es als sinnvoll, dieses Beispiel in C und in Java® zu implementieren.

Für COBOL II und REXX folgen anschliessend nur knappe Angaben zu den erforderlichen Strukturen und Kontrollen, da diese beiden Sprachen zwar einerseits nicht mehr die Bedeutung haben, die sie einst besaßen, andererseits aber immer noch weit verbreitet sind und vor allem in der kommerziellen Programmierung nach wie vor eine große Rolle spielen.

9.2.7 Implementierung in COBOL II

Die Definitionen für den „sports contest rec(ord)“ und für die Ranggruppenschlüssel lauten etwa wie folgt:

```
 77  FILLER                          PIC X(08)  VALUE 'SPORTREC'.
 01  SportsDSRec.
     05  Country-Code                PIC X(03).
     05  Region-ID                   PIC XX.
     05  School-No                   PIC X(08).
     05  FILLER                      PIC X.
     05  Gender-Code                 PIC X.
     05  FILLER                      PIC X.
     05  Age                         PIC 99.
     05  FILLER                      PIC X.
     05  Sprint-Points               PIC 99.
     05  FILLER                      PIC X.
     05  Jump-Points                 PIC 99.
     05  FILLER                      PIC X.
     05  Throw-Points                PIC 99.
     05  FILLER                      PIC X.
     05  Name                        PIC X(47).
*
 77  FILLER                          PIC X(08)  VALUE 'RANKKEYS'.
 01  Old-Rank-1-4-Key.
     05  Old-Rank-1-3-Key.
         10  Old-Rank-1-2-Key.
             15  Old-Rank-1-Key.
                 20  Read-Status     PIC X      VALUE 'D'.
                     88  End-Of-File            VALUE 'E'.
                 20  Country-Code    PIC X(03)  VALUE SPACES.
             15  Region-ID           PIC XX     VALUE SPACES.
             15  School-No           PIC X(08)  VALUE SPACES.
         10  Gender-Code             PIC X      VALUE SPACE.
     05  Age                         PIC 99     VALUE ZEROES.

*
 01  New-Rank-1-4-Key.
     05  New-Rank-1-3-Key.
         10  New-Rank-1-2-Key.
             15  New-Rank-1-Key.
                 20  Read-Status     PIC X      VALUE 'D'.
                     88  End-Of-File            VALUE 'E'.
                 20  Country-Code    PIC X(03)  VALUE SPACES.
             15  Region-ID           PIC XX     VALUE SPACES.
             15  School-No           PIC X(08)  VALUE SPACES.
         10  Gender-Code             PIC X      VALUE SPACE.
     05  Age                         PIC 99     VALUE ZEROES.
```

Hier fallen vor allem die Rangschlüsseltreppen ins Auge, bei denen die höheren Ränge in die niedrigeren geschachtelt zu sein erscheinen. Das ist aber nur ein optischer Eindruck. Tatsächlich ist das jeweils erste Byte der Rangschlüsselstruktur der `Read-Status`, gefolgt vom `Country-Code` auf demselben Level 20. Diese bilden zusammen den Rank-1-Key. Danach folgt der R-School-Group-Schlüssel = Rank-2-Key, der aus der Folge von Rank-1-Key, `Region-ID` und `School-No` besteht. Diese Konkatenation wird durch das unmittelbare Aneinanderreihen der beiden Felder auf dem Level 15 erzielt. `Gender-Code` kommt beim Rank-3-Key und `Age` beim Rank-4-Key hinzu.

Durch die Vergabe der höchsten Level-Nummern für den höchsten Rang (1), der zweithöchsten für den Rang 2 etcetera wird die Zusammenfassung der einzelnen Rangschlüssel zum Rank-1-2-Key, Rank-1-3-Key und Rank-1-4-Key bewirkt. Der neue `Read-Status` – nach dem Nachlesen – und die Rangschlüssel können auf sehr einfache Weise in den Kontrollen verwendet werden.

Hinweis: Hier ist zu beachten, dass Byte-Felder mit Level-Nummern > 01 von COBOL in unmittelbar aufeinanderfolgenden Speicherzellen angeordnet werden. Vergleiche konkatenierter Schlüssel finden daher physisch als Zeichenkettenvergleiche statt. Rangschlüsselelemente, die beispielsweise Ganzzahlen sind, werden aber vom jeweiligen Compiler gegebenenfalls an Halbwort-, Wort- oder Doppelwortgrenzen ausgerichtet, wobei der Inhalt eingeschobener Füll-Bytes undefiniert sein kann. Das würde auf jeden Fall zu ganz erheblichen Problemen führen. Es gibt aber noch weitere gute Gründe, keine anderen Datentypen als Zeichen- oder Ziffernketten in konkatenierte Schlüssel aufzunehmen, die z.B. aus den unterschiedlichen internen Zahldarstellungen resultieren.

Das Füllen der New-Rank-Key-Felder aus den `SportsDSRec`-Inhalten nach dem Lesen ist so einfach, dass es hier nicht gezeigt werden muss. Dasselbe gilt für das Kopieren der neuen Rangschlüssel unmittelbar vor den Kontrollen I2 bis I6. Die implementierten Iterationskontrollen I1 bis I6 würden lauten:

```
I1:     PERFORM UNTIL End-Of-File
           …
        END-PERFORM

I2:     PERFORM UNTIL New-Rank-1-Key > Old-Rank-1-Key
           …
        END-PERFORM

I4:     PERFORM UNTIL New-Rank-1-2-Key > Old-Rank-1-2-Key
           …
        END-PERFORM

I5:     PERFORM UNTIL New-Rank-1-3-Key > Old-Rank-1-3-Key
           …
        END-PERFORM

I6:     PERFORM UNTIL New-Rank-1-4-Key > Old-Rank-1-4-Key
           …
        END-PERFORM
```

COBOL kann Zahlen, die als Ziffernfolgen dargestellt sind, für Rechenoperationen intern konvertieren. Das gilt auch für die Punktanzahlen im `SportsDSRec`. Daher ist es hier nicht zwingend nötig, die Struktur SpoConD explizit zu definieren und zu befüllen. Aus der Performance-Sicht sollte man interne Konvertierungen nicht öfter als unvermeidbar vornehmen lassen, da das Rechnen mit Ziffernfolgen wegen der ständig wiederholten

Abbildungen in die duale Darstellung und zurück höchst ineffizient ist. Die einmalige Addition der erzielten Punkte pro „sports contest rec(ord)“ lässt sich aber durch eine explizite Konvertierung vor der Addition nicht schneller machen.

Die COBOL-II-Implementierung der übrigen Operationen und Kontrollen ist im aktuellen Kontext uninteressant und wird daher hier nicht beschrieben.

9.2.8 Implementierung in REXX

Da REXX keine echten Deklarationen kennt, müssen Rangschlüssel dort durch disziplinierte Verwendung der Schlüsselelemente und -abschnitte nachgeahmt werden.

a) Code in der Lese-Routine einer Implementierung unter TSO und z/OS® (Op. 3)

```
Address TSO "EXECIO 1 DISKR SPORTREC (STEM SportsDSRec."

If SportsDSRec.0 > 0 Then Do
   new_country_code = Substr(SportsDSRec.1, 01, 03)
   new_region_ID    = Substr(SportsDSRec.1, 04, 02)
   new_school_no    = Substr(SportsDSRec.1, 06, 08)
   new_gender_code  = Substr(SportsDSRec.1, 15, 01)
   new_age          = Substr(SportsDSRec.1, 17, 02)
   read_status      = 'D'
   End
Else
   read_status      = 'E'

new_rank_1_4_key    = read_status     !! new_country_code !! , /* Rank 1 */
                      new_region_ID   !! new_school_no    !! , /* Rank 2 */
                      new_gender_code !!                     , /* Rank 3 */
                      new_age                                  /* Rank 4 */
new_rank_1_3_key    = read_status     !! new_country_code !! , /* Rank 1 */
                      new_region_ID   !! new_school_no    !! , /* Rank 2 */
                      new_gender_code                          /* Rank 3 */
new_rank_1_2_key    = read_status     !! new_country_code !! , /* Rank 1 */
                      new_region_ID   !! new_school_no         /* Rank 2 */
new_rank_1_key      = read_status     !! new_country_code
```

Anmerkungen

- Folgende Initialisierungen sind am Programmanfang sinnvoll:

```
new_country_code = '   '
new_region_ID    = '  '
new_school_no    = '        '
new_gender_code  = ' '
new_age          = '  '
```

- Die Nutzung von vier verschiedenen Rangschlüsselvariablen erscheint auf den ersten Blick als exzessiv. Die Alternative wäre jedoch, mit Substrings zu arbeiten, was unübersichtlich, ineffizient und – wegen der starren Zeichenpositionen – wartungsunfreundlich wäre. Beispiel:

```
old_rank_1_key = Substr(new_rank_1_4_key, 01, 04)       /* op. 08 */
countries      = countries + 1                          /* op. 13 */
schools        = 0                                      /* op. 14 */

I2:     Do Until Substr(new_rank_1_4_key, 01, 04) > old_rank_1_key
           …
        End
```

- Anstatt einzelner `new`-Variablen könnte auch ein Stem für alle Schlüsselelemente verwendet werden. Dadurch ist aber keine wesentliche Vereinfachung oder bessere Übersichtlichkeit erreichbar. Daher reichen die Präfixe `new` und `old` völlig aus.
- Es besteht kein Bedarf, die bisherigen Schlüsselelemente einzeln aufzubewahren, weil nur mit den konkatenierten Rangschlüsseln gearbeitet wird. Deshalb sind `old`-Einzelwertzuweisungen zusätzlich zu den Operationen 8 bis 11 nicht erforderlich. Somit entfallen auch `old`-Initialisierungen.
- REXX arbeitet aus Programmierersicht beim Rechnen sowieso nur mit Ziffernstrings (gegebenenfalls mit Dezimalpunkt und Vorzeichen). Daher ist auch hier die Struktur SpoConD ohne Bedeutung.

Die implementierten Iterationskontrollen I1 bis I6 würden wie folgt lauten:

```
I1:     Do Until read_status = 'E'
           …
        End

I2:     Do Until new_rank_1_key > old_rank_1_key
           …
        End

I4:     Do Until new_rank_1_2_key > old_rank_1_2_key
           …
        End

I5:     Do Until new_rank_1_3_key > old_rank_1_3_key
           …
        End

I6:     Do Until new_rank_1_4_key > old_rank_1_4_key
           …
        End
```

Der sonstige REXX-Code ist hier nicht weiter von Interesse.

9.3 Hinzunahme der Schul-Daten

Da die Schul-Daten, also Name und (E-Mail-)Adresse nicht in der Eingabedatei „Sports Dataset" enthalten sind, müssen sie aus einer zweiten Datenquelle beigesteuert werden. Bevor wir uns den dafür möglichen Optionen zuwenden, lassen Sie uns ein Gedankenexperiment machen, in dem es keine solche zweite Datenquelle gibt. Wie soll das gehen?

9.3.1 Zeitreise zurück in die 60er Jahre

Wir kehren in die „Steinzeit" der Datenverarbeitung zurück und stellen für die vorliegende Aufgabe lediglich folgende Konfiguration zur Verfügung:

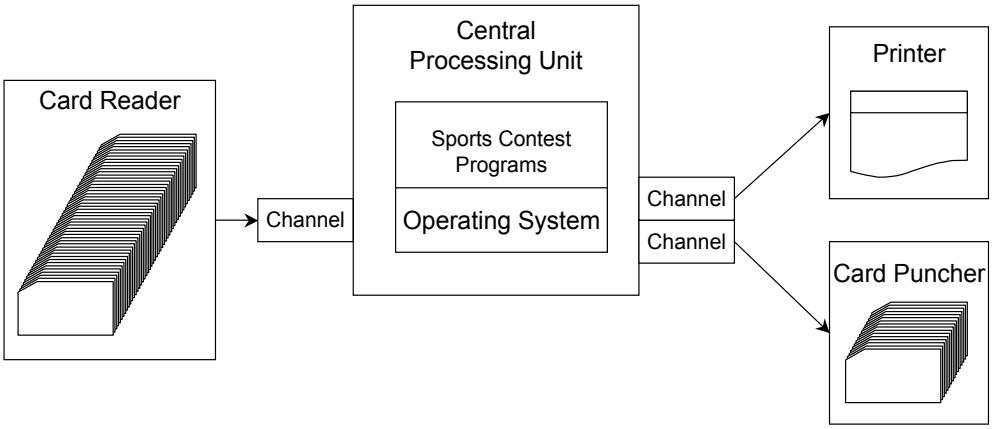


Abbildung 9.16: *Fiktive Konfiguration mit Kartenleser, -stanzer und Drucker*

Es gibt also nur die Eingabedatei, das ist ein per Kartenleser einzulesenden Kartenstapel, die Druckdatei – für das Protokoll – und einen Kartenstanzer als Ausgabedatei, denn die Programmlauf-Ausgaben sollen ja sortiert und weiterverarbeitet werden. Kein Bandgerät. Kein Plattenspeicher. Erst recht keine Datenbank. Was nun?

Auch dann könnte die Auswertung der Schulergebnisdaten stattfinden und das gewünschte Resultat liefern. Ohne zweites Eingabegerät – zweiter Kartenleser, Bandgerät, … – können wir die Schul-Daten jedoch nicht wie gewünscht beschaffen und in die Ausgabe des Programms integrieren. Wir müssen uns anders behelfen.

Für solche und ähnliche Zwecke wurden die „Satzarten" erfunden. Bislang kennen wir ja nur die „sports contest records", die in monotoner Folge sortiert aufeinanderfolgen. Diese sind 28 + 47 = 75 Zeichen lang. Eine Standard-Lochkarte hat(te) 80 Zeichen, also haben wir noch genug Zeichenpositionen für besondere Zwecke übrig. Traditionell werden dafür die Zeichen am Anfang oder Ende der Karten verwendet. Dann kann eine Eingabedatei, die auch die Schul-Daten – natürlich ohne E-Mail-Adressen – enthält, wie folgt aufgebaut sein:

```
0 DEUBW00470811 HOELDERLIN-GYMNASIUM
1 DEUBW00470811 88214 RAVENSBURG, BADEN-WUERTTEMBERG, DEUTSCHLAND
2 DEUBW00470811 F 10 08 09 06 LEIBEL, MARIA
2 DEUBW00470811 F 10 11 08 12 RUSSO, ANTONELLA
[…]
2 DEUBW00470811 M 19 12 13 08 HAGELUND, KLAUS
2 DEUBW00470811 M 19 15 10 11 PFÄLZER, GERHARD
```

```
[…]
0 ITASI45008132 LICEO FEDERICO FELLINI
1 ITASI45008132 92014 PORTO EMPEDOCLE, SICILIA, ITALIA
2 ITASI45008132 F 15 15 16 07 COGLITORE, LAURA
2 ITASI45008132 F 15 04 07 06 PERITORE, MARIUCCIA
2 ITASI45008132 F 16 03 07 06 DI STEFANO, SERENA
2 ITASI45008132 F 16 03 09 08 LO PORTO, MARIUCCIA
2 ITASI45008132 F 16 18 15 17 PRESTIA, ANGELA
2 ITASI45008132 F 17 10 08 12 CASTAGNOLA, MICHELA
[…]
```

und so weiter bis zum Dateiende.

Das neue Format der „sports contest records" ist dann (mit vorangestelltem Positions-Lineal):

```
         11111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
T CCCRRSSSSSSSS G AG RN LJ LT NAME (length <= 47)
```

Hinweise: Das `'T'` auf der ersten Position bedeutet „Record Type" = Satzart. Daran kann das Programm ablesen, welche Bedeutung die Daten auf der jeweiligen Karte besitzen. 1960 mussten alle Buchstaben groß und ohne Umlaute / Accents etcetera geschrieben werden.

Sortieren der Eingabedatei

Zusätzlich dienen die absichtlich in der Folge 0 – 1 – 2 vergebenen Satzarten als Sortierkriterium. Angenommen, die Eingabesätze müssten zunächst sortiert werden, wobei wir für einen Augenblick in die Gegenwart zurückkehren und das Vorhandensein einer modernen Massenspeicherperipherie, sowie eines leistungsfähigen Sortierprogramms unterstellen. Dieses Programm benötigt Steuerungsdaten, die festlegen, was – welche Bytes in jedem Satz – in welcher Folge – wahlweise aufsteigend oder absteigend – sortiert werden soll. Diese Angaben heissen bei etlichen Sortierprogrammen `SORT FIELDS`.

Damit die Sätze in dieser Datei nach der Sortierung korrekt aufeinanderfolgen, müssen die bisherigen Sortierkriterien geändert werden. An die Stelle der Feldfolge `CCC,RR,SSSSSSSS,G,AG`

⇨ `SORT FIELDS=(01,18,CH,A)`

beim bisherigen Satzaufbau tritt die Feldfolge `CCC,RR,SSSSSSSS,T,G,AG`

⇨ `SORT FIELDS=(03,13,CH,A,01,01,CH,A,17,04,CH,A)`

beim neuen Format. Dabei werden Leerzeichen aus Bequemlichkeitsgründen in die Sortierung einbezogen. Dann kann nämlich der 18-stellige Satzpräfix beim bisherigen Format komplett sortiert werden. Beim neuen Format ist nur das Leerzeichen auf der Position 18 dabei.

Die Syntax der `SORT FIELDS` ist sehr einfach. Im Fall `(01,18,CH,A)` sollen ab der Position 01 einschliesslich 18 Zeichen = Characters (`CH`) aufsteigend = ascending (`A`) sortiert werden. Genauer gesagt, werden natürlich nicht nur diese Zeichen sortiert. Das Sort-Programm extrahiert die beiden Felder aus jedem Satz und ordnet die Sätze – ohne Änderung an deren Inhalten – so um, dass sie in der Ausgabe die gewünschte Folge bilden.

Im zweiten Fall muss wegen der neuen Sätze nach 13 Zeichen (die auf der Position 03 beginnen), die Satzart auf der ersten Position eingeschoben werden. Dadurch gelangen die Schul-Daten an den jeweiligen Rang-1-2-Anfang, unabhängig vom Inhalt der Gender-Code- und Age-Felder. Diese enthalten bei den Satzarten 0 und 1 sowieso nichts Brauchbares.

Anstelle von Zeichen (CH) können auch andere Datentypen angegeben werden, zum Beispiel PD (= packed decimals), ZD (= zoned decimals), BI (= binary) – gegebenenfalls mit bitgenauer Vorgabe, FL (= normalized hexadecimal floating-point data). Die Alternative zu „ascending" ist „descending" (D).

Diese zeichen- und bitpositionsorientierten Kleinlichkeiten erscheinen in der schönen neuen Multimedia-Welt als völlig antiquiert. Sie haben aber bei allen Verarbeitungen, die externe Sortierungen erfordern, nach wie vor eine hohe Bedeutung. Ein scheinbar kleiner Sortierfehler kann dann dramatische Folgen haben. Daher muss jede(r) kommerzielle Programmierer(in) sich damit gut auskennen.

Heutzutage spielt die 80-Zeichen-Grenze keine wichtige Rolle mehr. Sätze mit vielen Sortierfeldern und auch mit Satzarten dürfen weitaus länger sein, obwohl es auch hier physische Grenzen gibt, die je nach Dateiorganisation beispielsweise maximal knapp 32 KB pro Satz erlauben. Programmcode wird dagegen – zumindest auf Großrechnern – oft noch in Dateien aufbewahrt, deren feste Satzlänge 80 Bytes beträgt. Diese Sätze werden normalerweise geblockt, also zu Blöcken zusammengefasst, die eine gute Platznutzung der Plattenmedien ermöglichen.

Am Rande bemerkt: Die über Jahrzehnte zu höchster Rafinesse ausgefeilten Sortier-Dienstprogramme bewältigen heute Gigabytes (10^9 Bytes) oder Terabytes (10^{12} Bytes), irgendwann aber auch Petabytes (10^{15} Bytes). Sie bieten eine Fülle von Fähigkeiten, die über das reine Sortieren weit hinausgehen. Wer sich diese Fähigkeiten zunutze macht, kann oftmals ganz ohne eigene Programmierung arbeiten, oder diese durch Nutzung von Sort-Exits auf das Unvermeidliche beschränken - und dabei um vieles effizienter als mit einem Anwendungsprogramm operieren.

Verarbeitung des Kartenstapels

Wir gehen wieder in die Vergangenheit zurück und füllen die wohlsortierten Karten mit den Schul-Daten und den „sports contest records" in den Eingabeschacht des Kartenlesers. Ein Tastendruck verursacht einen Interrupt im Prozessor. Dieser startet den Kartenleser und damit letztlich das Anwendungsprogramm. Das wäre um 1960 wahrscheinlich ein Assembler-Programm gewesen.

Ein relativ kleiner Speicherbereich würde völlig für die Laufzeitdaten des Programms ausreichen, da ja nur die kurzen Siegerlisten und die zugehörigen Schul-Daten längerfristig gespeichert werden müssen. Der erzeugte Kartenstapel wäre wesentlich kürzer als der Eingabestapel, weil er pro Schule selten länger ist als 6 Karten. Dazu kommen die Karten für die nationalen und internationalen Sieger.

Der Ausgabestapel muss sortiert werden, um die Sieger-Karten von hinten nach vorn zu bewegen, wobei die internationalen Sieger ganz vorn und die Landessieger jeweils vor den Landesergebnissen einsortiert werden müssen. Die Feldfolge und die Sortierkriterien lauten:

```
CCC,RR,SSSSSSSS,LN  ⇨      SORT FIELDS=(01,15,CH,A)
```

(mit LN = Line Number). 1960 war es durchaus möglich, dass in einem Rechenzentrum für eine solche Sortierung keine Bandgeräte als Zwischenspeicher zur Verfügung standen. Stattdessen behalf man sich mit elektromechanischen Sortiermaschinen, die auf den Erfinder Herman Hollerith und das Ende des 19. Jahrhunderts zurückgehen.

Die sortierten Karten sind die Eingabe für ein weiteres Programm, das daraus damals die Ergebnislisten erzeugt und auf Papier mit einem Seitenvorschub vor jeder Schule gedruckt hätte. Danach: Postversand der Ergebnisblätter. Keine E-Mail. Kein Internet-Zugang zu einem Server mit den Ergebniszusammenstellungen.

Dennoch hätte der Internationale Schulsportwettbewerb stattfinden können. Der Auswertungsalgorithmus wäre seitdem weitgehend derselbe geblieben, aber immer mal wieder in einer neueren Sprache codiert worden.

Was wurde eigentlich aus den Lochkarten? Hierzu ein Zitat aus Wikipedia[3]:

»**Ende der Lochkarten-Ära**
Ab Mitte der 1960er Jahre verbreiteten sich in den Rechenzentren Magnetbänder als Medium zum Speichern und Sortieren von Daten. Sie waren schneller und hatten eine wesentlich höhere Kapazität in Bezug auf Volumen und Gewicht. Mitte der 1970er Jahre war die Lochkarte so gut wie nicht mehr existent und auch zur Datenerfassung zum Beispiel von Magnetbandkassetten und/oder Disketten abgelöst. Dieser technologischen Entwicklung entsprechend verschwanden auch Lochkartensortierer aus den Rechenzentren.«

9.3.2 $10^3 - 10^6 - 10^9 - 10^{12} - 10^{15} - 10^{18} - 10^{21} - 10^{24} - \ldots$

Wozu soll dieser Ausflug in die Vergangenheit dienen? Das werden Sie sich vielleicht gefragt haben. Uns steht doch heutzutage eine Fülle technischer Möglichkeiten zur Verfügung, an die damals noch gar nicht zu denken war. Plattenplatz und Rechenkapazität sind anders als früher keine ernstzunehmenden Restriktionen mehr. Warum sich den Kopf über Performancefragen zerbrechen, wo doch noch jede Rechnergeneration einen enormen Leistungsschub mit sich brachte?

Das ist richtig – und falsch zugleich. Für die Benutzer von gut ausgestatteten Rechnerkonfigurationen scheinen die früher üblichen Einschränkungen und Engpässe kaum mehr zu existieren. Vielleicht ist das der Grund dafür, dass immer mehr Programmierer(innen) mit den Betriebsmitteln hausen, anstatt sparsam damit umzugehen. Den Dauerbrenner „SQL-Optimierung" müsste es eigentlich gar nicht geben, wenn jede(r), der oder die mit SQL arbeitet, die Folgen seines oder ihres Tuns genau verstehen und sich danach richten würde. Die wachsende Komplexität der Datenbanksysteme, die meist völlig unterentwickelte innerbetriebliche Fortbildung und der Druck, rasch funktionierenden Code zu produzieren, machen es aber zunehmend schwer, diesen löblichen Maximen zu folgen.

Aufgrund der enormen Fortschritte, die auf dem Gebiet der Rechnerarchitekturen, sowie bei den Verarbeitungs- und Transfergeschwindigkeiten erreicht wurden, scheinen die anhaltenden Bemühungen um bessere Performance zunehmend sinnlos zu werden. Wenn es nur noch eine Sekunde dauert, 10 Megabyte zu verarbeiten, wozu sich um eine Halbierung dieser Sekunde bemühen und dafür womöglich mehrere Programmiertage aufwenden? Das ist für eine einzelne Verarbeitung von 10 MB am Tag leicht zu beantworten: völlig sinnlos. In einem Großrechnerverbund, der täglich Millionen von Programmläufen oder Transaktionen abwickelt, führt die Verschwendung von Ressourcen in zahlreichen Einzelfällen jedoch zu einer erheblich höheren Gesamtlast, und die kostet Geld und Zeit.

Im Grunde haben sich in den vergangenen Jahrzehnten nur die technologischen Herausforderungen und Grenzen nach oben verschoben. Wo vor einiger Zeit noch ein Megabyte eine respektierte Größenordnung war, vor sehr viel kürzerer Zeit dann ein Gigabyte, wachsen die Bestände nun im Terabytebereich. Es kann nicht mehr lange dauern, bis wir zu den Petabytes vorstoßen, die wir uns nicht mehr vorstellen können, ganz zu schweigen von Exabytes (10^{18} Bytes), Zettabytes (10^{21} Bytes) und Yottabytes (10^{24} Bytes). Gleichzeitig begleitet uns stets die Frage: Sind unsere Programme schnell genug?

Wenn 10 Megabytes (10^7 Bytes) in einer Sekunde gegen eine Datenbank verarbeitet werden können, so werden für die höheren Größeneinheiten folgende Zeiten benötigt:

- 1 Gigabyte: 10^2 Sekunden = 1 Minute, 40 Sekunden
- 1 Terabyte: 10^5 Sekunden = 27 Stunden, 4 Minuten, 40 Sekunden
- 1 Petabyte: 10^8 Sekunden = 1.157 Tage, 9 Stunden, 46 Minuten, 40 Sekunden

Selbst bei massivem Einsatz paralleler Rechenkapazitäten sind die noch höheren Einheiten derzeit unerreichbar weit von realistischen Zeitmaßen entfernt.

Das untere Ende der Rechnerskala markieren die Mikrocontroller, die mehr und mehr für elektronische und elektromechanische Steuerungszwecke eingesetzt werden. Das sind auch Computer, aber völlig andere als die Universalrechner, mit denen der Aufschwung der Rechenzentren begann. Mikrocontroller besitzen Rechenfähigkeiten, können aber auch elementare Schaltvorgänge auslösen, Sensoren und Uhren-Chips abfragen, digitale Signalprozessoren oder Interface-Bausteine für die Komponentenkommunikation unterstützen und vieles mehr. USB- und TTL-Schnittstellen sind ebenfalls üblich (TTL = Transistor-Transistor-Logik). Damit lassen sich hochinteressante Anwendungen entwickeln, die unseren Alltag voraussichtlich weit mehr revolutionieren werden, als dies die herkömmliche Rechnertechnologie inklusive Internet bereits getan hat.

Genau auf dieser Ebene treffen wir die Kargheit der 60er Jahre wieder an. Im schlimmsten Fall müssen sogar Interrupts durch das jeweilige Anwendungsprogramm erkannt und behandelt werden, denn es gibt bei den kleinsten Mikrocontrollern kein Betriebssystem (OS = Operating System). Um elementare Schaltvorgänge auszuführen und Sensoren abzufragen, ist kein OS erforderlich. Mikrocontroller mit höherer Leistungsfähigkeit können durchaus ein Real-Time-OS beherbergen, das mit dem Code der Anwendung zusammen geladen wird, oder sogar schon beim ersten Einschalten präsent ist. Inwischen sind seit Jahren Webserver auf Linux®-Basis verfügbar, die in einem Ethernet-Stecker Platz finden. C- und Java-Compiler, sowie die erforderlichen Laufzeitumgebungen stehen ebenfalls für Microcontroller und Mikroprozessoren zur Verfügung.

Welche Minimalkonfiguration wäre also 2022 für den Internationalen Sportwettbewerb ausreichend?

9.3.3 Technische Minimalkonfiguration 2022

Es lohnt sich immer wieder, sich zu überlegen, mit welchen Betriebsmitteln – Rechner und Peripherie – eine bestimmte Anwendung minimal zurechtkommen sollte. Spätestens dann, wenn wieder einmal ein Downsizing verlangt wird, oder die Anwendung eine neue Heimat finden muss, ist es gut, darüber im Klaren zu sein.

Was brauchen wir im vorliegenden Fall minimal?

- Eine CPU, beispielsweise einen Mikroprozessor
- Ein geeignetes Betriebssystem, auf dem C- oder Java-Programme ausgeführt werden können
- Hauptspeicher für das Betriebssystem, die Library-Routinen und das jeweilige Anwendungsprogramm
- Einen USB-Stick, auf dem die Eingabedaten – Schul-Daten und „sports contest records" getrennt – sortiert gespeichert sind, und auf den die Ausgabedaten des Auswertungsprogramms geschrieben werden
- Ein Sortierprogramm, das diese Ausgabedaten für den Versandlauf umstellt
- Ein Programm, das die sortierten Daten in E-Mails umsetzt und dem Client zum Versand übergibt
- Einen E-Mail-Client und dessen Zugang zu einem Provider
- Einen Ethernet-Anschluss für die E-Mail-Daten (an ein Netzwerk mit Internet-Zugang)

Das alles kann auf etlichen Kubikzentimetern untergebracht werden. Die 1960 erforderliche Hardware ist also um circa 2 Zehnerpotenzen in jeder der drei Dimensionen geschrumpft. Wir sollten allerdings nicht vergessen, dass diese Minimalkonfiguration ein Entwicklungssystem voraussetzt, das es ermöglicht, den Mikroprozessor zu programmieren. Dafür reichen jedoch wenige Quadratdezimeter – inklusive Laptop – aus.

Eine solche Minimalkonfiguration könnte wie in der „Abbildung 9.17: Fiktive Minimalkonstellation auf Mikroprozessorbasis mit USB und Ethernet" auf Seite 230 aussehen. Die Daten auf dem USB-Stick sind hier nur angedeutet. Dort könnten auch Programm- und Library-Files etcetera liegen. Die von den Programmen auf dem USB-Stick abgelegten Laufprotokolle können anschliessend an einem PC betrachtet und, falls erforderlich, gedruckt werden.

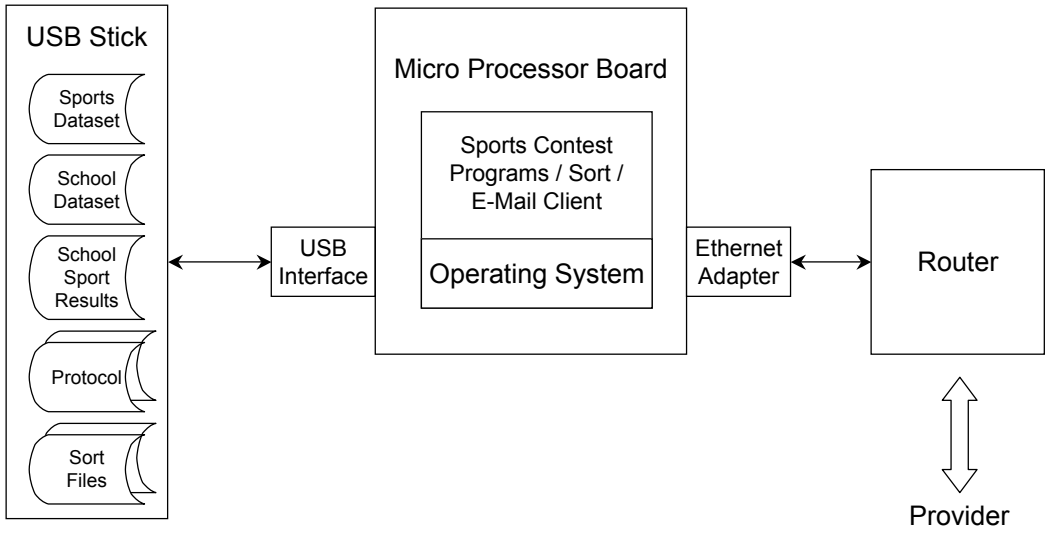


Abbildung 9.17: *Fiktive Minimalkonstellation auf Mikroprozessorbasis mit USB und Ethernet*

Nun erscheint es doch als reichlich aufwendig, für eine solche Aufgabe extra einen Mikroprozessor samt Peripherie anzuschaffen und zu programmieren. Ein gewöhnlicher PC reicht hierfür völlig aus. Dennoch zeigt dieses Gedankenexperiment, in welche Richtung sich Technologien entwickeln können, an die vor Jahrzehnten nicht in den wildesten Träumen zu denken war.

9.3.4 Schul-Daten in einer zweiten Datei

Nachdem wir uns über die technischen Voraussetzungen klargeworden sind, können wir uns wieder dem Eingangsthema zuwenden. Auf einem USB-Stick oder einer Festplatte sind problemlos viele Dateien unterzubringen. Eine davon soll die sortierten „sports contest records" im bisherigen Format und eine weitere die in derselben Folge sortierten Schul-Daten enthalten. Um Längen- und Ausrichtungsproblemen aus dem Weg zu gehen, sind die Daten zu jeder Schule in jeweils drei Sätzen zu – maximal – 80 Zeichen Länge untergebracht.

Das Satzformat dieser Schul-Daten sei (mit vorangestelltem Positions-Lineal):

```
         11111111112222222222333333333344444444445555555556666666666777777777 8
12345678901234567890123456789012345678901234567890123456789012345678901234567890
CCCRRSSSSSSSS01 SCHOOL NAME             (name length <= 50)
CCCRRSSSSSSSS02 SCHOOL ADDRESS          (length <= 64)
CCCRRSSSSSSSS03 SCHOOL.NAME@PROVIDER.XY (length <= 64)
```

Da wir von geprüften Daten ausgehen dürfen, setzen wir voraus, dass zu jedem Schulschlüssel einer „sports contest records"-Ranggruppe genau eine dazu synchrone Satzgruppe existiert, welche die zugehörigen Schul-Daten enthält. Wir müssen also an jedem Rang-2-Beginn drei Sätze mit Schul-Daten lesen und verarbeiten. Daraus ergeben sich folgende Korrespondenzen zwischen den Schul-Daten, dem Sports Dataset und den School Sport Results.

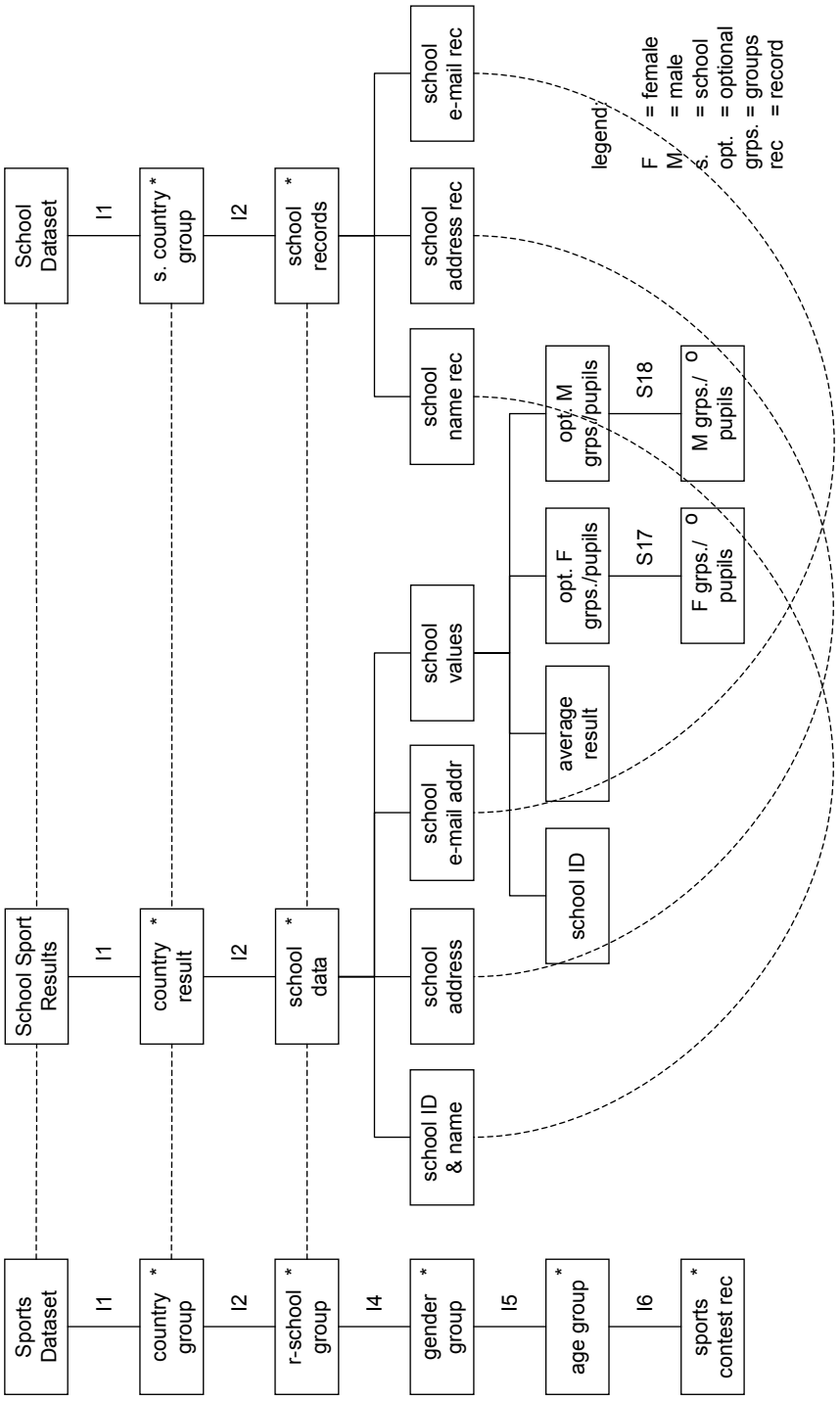


Abbildung 9.18: *Korrespondenzen unter Einbeziehung der synchronen Schul-Daten*

Durch Verschmelzen der korrespondierenden Strukturblöcke miteinander und durch deren Übertragung in die Programmstruktur ohne Operationen entsteht die folgende erweiterte Programmstruktur:

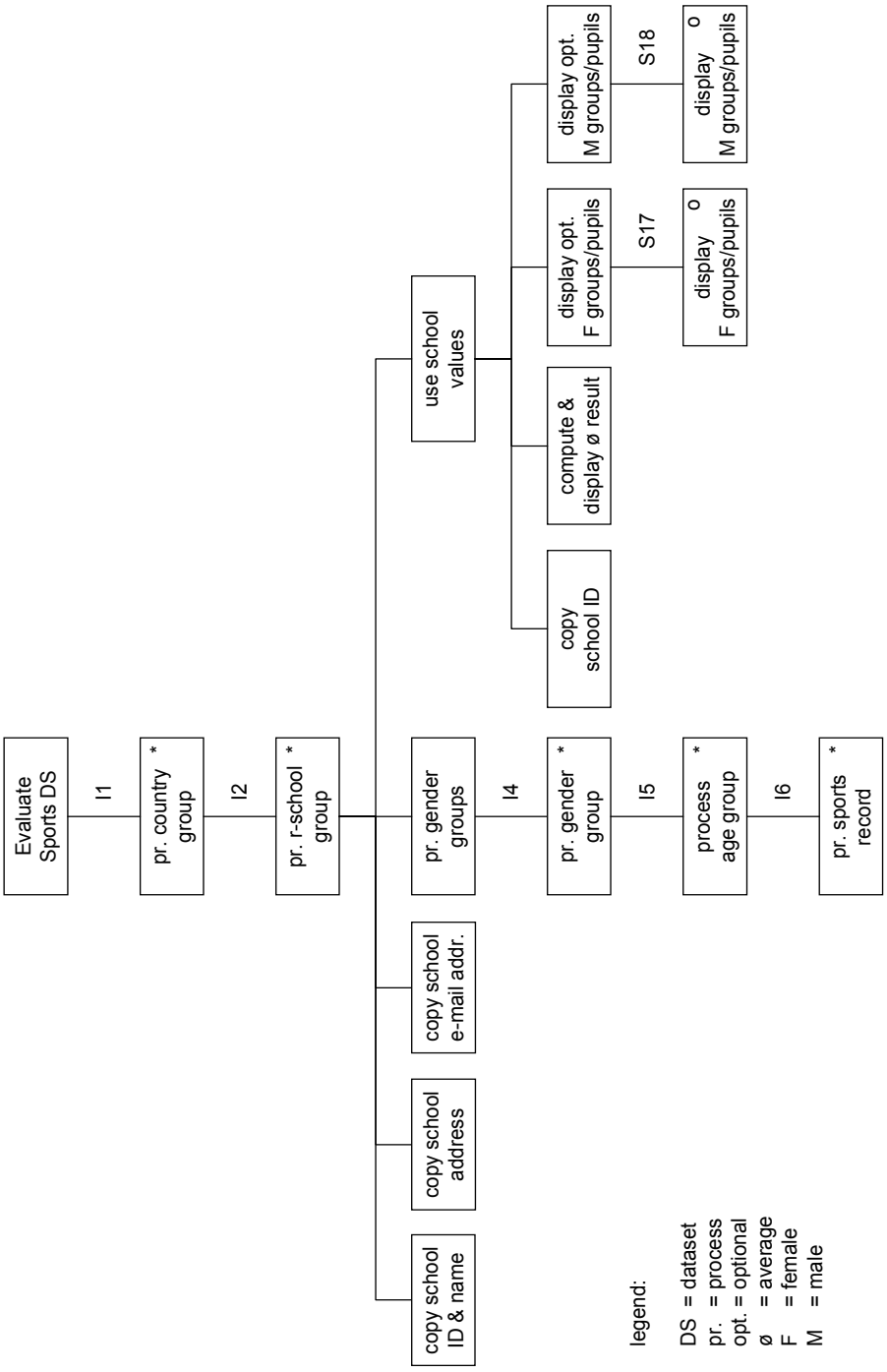


Abbildung 9.19: *Erweiterte Programmstruktur mit Verarbeitung der Schul-Daten*

Nun können wir auch den zweiten Anteil der geplanten Ausgabedatei produzieren. Wegen der – noch zu erzeugenden – internationalen Ergebnisse muss die School ID ab der Position 17 der Zeile 01 wiederholt werden. Erst dann folgt – nach einem Leerzeichen – der Schul-Name. Die Ausgabe sieht jetzt wie folgt aus:

```
DEUBW0047081101 DEUBW00470811 Hölderlin-Gymnasium
DEUBW0047081102 88214 Ravensburg, Baden-Württemberg, Deutschland
DEUBW0047081103 sekretariat@hoelderlin.rv.bw.schule.de
DEUBW0047081104 Ø 33,50 F 4/129 M 7/432
[…]
DEUBW0765432101 DEUBW07654321 Paul-Klee-Realschule
DEUBW0765432102 70565 Stuttgart, Baden-Württemberg, Deutschland
DEUBW0765432103 paul-klee-schule@stuttgart.de
DEUBW0765432104 Ø 28,70 F 5/305 M 6/364
[…]
DEUBY0012345601 DEUBY00123456 Ludwig-Thoma-Gymnasium
DEUBY0012345602 86899 Landsberg am Lech, Bayern, Deutschland
DEUBY0012345603 sekretariat@ltg-landsberg.de
DEUBY0012345604 Ø 33,65 F 4/372 M 6/522
[…]
DEUHE0022334401 DEUHE00223344 Immanuel Kant Gymnasium
DEUHE0022334402 60322 Frankfurt am Main, Hessen, Deutschland
DEUHE0022334403 schulleitung@kant-gymnasium.frankfurt.de
DEUHE0022334404 Ø 34,32 F 6/451 M 6/562
[…]
DEUNW0001705001 DEUNW00017050 Schiller-Gesamtschule
DEUNW0001705002 48147 Münster, Nordrhein-Westfalen, Deutschland
DEUNW0001705003 schillerschule@muenster.de
DEUNW0001705004 Ø 31,28 F 7/642 M 8/583
[…]
ESPAN0278549101 ESPAN02785491 Instituto Federico García Lorca
ESPAN0278549102 18000 Granada, Andalucia, España
ESPAN0278549103 info@ifgl-granada.com
ESPAN0278549104 Ø 36,74 F 3/273 M 3/481
[…]
FRA530011276501 FRA5300112765 Licée Paul Signac
FRA530011276502 35800 Saint-Briac-sur-Mer, Bretagne, France
FRA530011276503 licee-signac@mairie-saint-briac.fr
FRA530011276504 Ø 36,65 F 5/284 M 7/462
[…]
ITALI8300785201 ITALI83007852 Liceo Scientifico Galileo Galilei
ITALI8300785202 18100 Imperia, Liguria, Italia
ITALI8300785203 segreteria@galilei-imperia.org
ITALI8300785204 Ø 35,75 F 3/147 M 5/321
```

Damit kommen wir unserem Ziel bereits ein schönes Stück näher. Die neuen Operationen und die ergänzte Programmstruktur folgen auf der nächsten Seite.

Op.	Bedeutung	Kommentar
31	School-Dataset öffnen	txt-Datei wird erwartet
32	School-Record lesen, konkat. Gruppenschlüssel zuweisen	⇨ school_rec
33	School-Dataset schliessen	
34	RR01 := Substr(school_rec, 0, 16) + Substr(school_rec, 0, 13) + ' ' + Substr(school_rec, 16, 50)	school ID + '01 ' + school ID + Blank + School Name (Start-Index von school_rec = 0)
35	Result-Satz 01 (RR01) ausgeben	
36	school_address := Substr(school_rec, 16, 64)	für Ranking aufbewahren
37	school_rec ausgeben	gleiches Format für Sätze 02 und 03

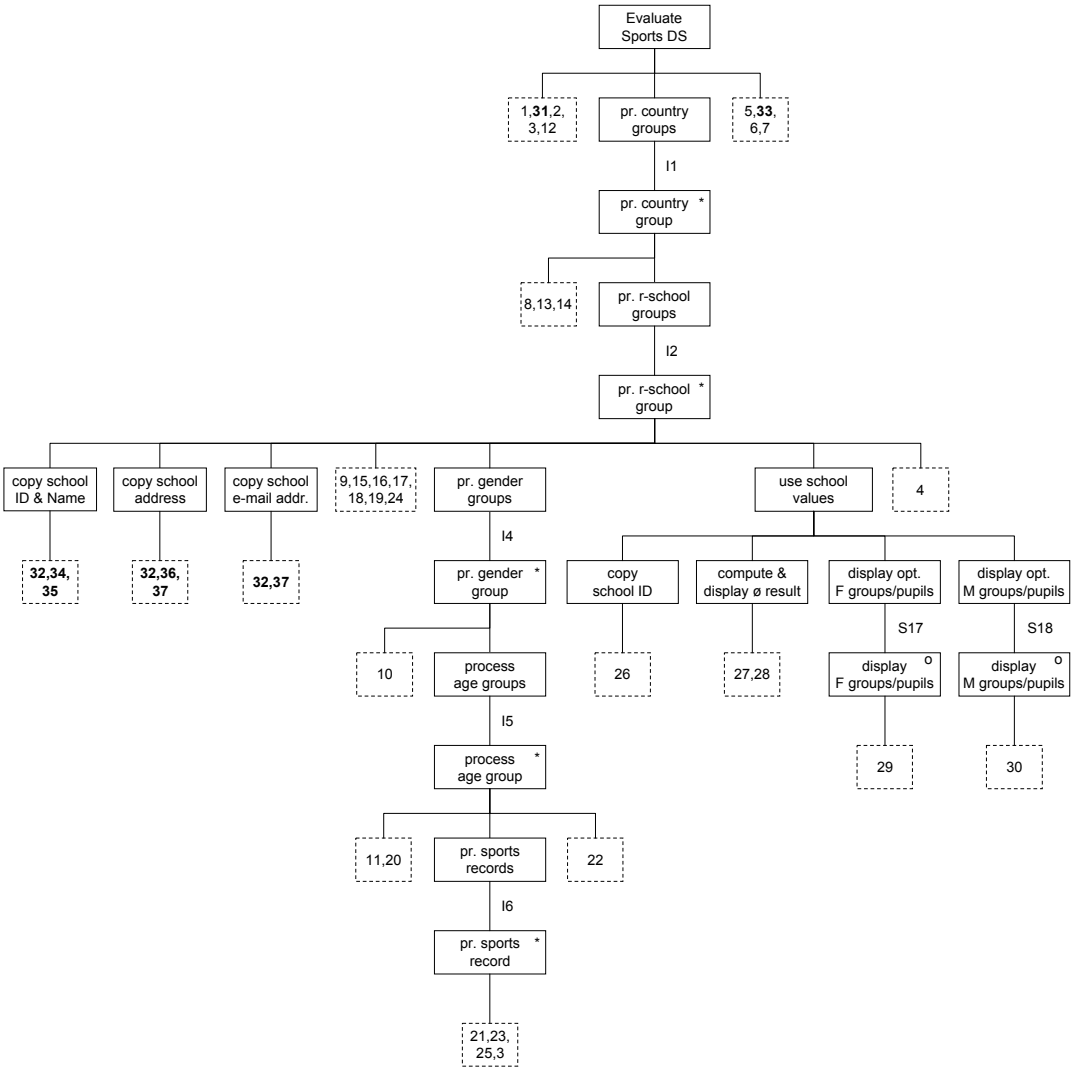


Abbildung 9.20: *Erweiterte Programmstruktur mit den neuen Operationen*

Die neuen Operationen sind fett markiert.

Der dritte und letzte Anteil der geplanten Ausgabedatei stammt aus der Ermittlung der Medaillenränge für Schüler und Schulen. Die Vergabe von Gold, Silber und Bronze auf drei Gruppenebenen erfolgt nach einem eigenen Verfahren, das im Kapitel „10 Link-Listen-Verarbeitung" auf Seite 237 ff entworfen wird. Das Kapitel „11 Ranggruppenverarbeitung II" auf Seite 291 ff führt die beiden Lösungsstränge zusammen und präsentiert die Gesamtlösung.

Endnoten

1 Die **Maison Carrée** (deutsch: Rechteckiges Haus) in Nîmes, Frankreich, ist einer der am besten erhaltenen Tempel auf dem Gebiet des Römischen Reiches.

Er wurde ganz zu Beginn des 1. Jahrhunderts n. Chr., möglicherweise auf Veranlassung des Marcus Vipsanius Agrippa errichtet. Der Tempel war den Söhnen des Agrippa, Gaius und Lucius, den jung verstorbenen Adoptivsöhnen des Augustus, gewidmet. […]

Die Maison Carrée ist ein hervorragendes Beispiel eines klassischen augusteischen Podiumstempels. Sie erhebt sich auf einem 2,85 m hohen Podium, das das Forum der römischen Stadt überragte. Die rechteckige Grundfläche ist beinahe zweimal so lang wie breit (26,42 m mal 13,54 m). Die Vorderseite des pseudoperipteralen Tempels wird von einer tiefen prostylen Vorhalle dominiert, die ein Drittel der Gebäudelänge einnimmt. Ihre zehn Säulen besitzen wie die 20 Halbsäulen der äußeren Cellawände korinthische Kapitelle. Über dem auf den Kapitellen liegenden Drei-Faszien-Architrav folgt ein Fries mit feinen Reliefs, die Rosetten und Akanthusblätter zeigen. Eine große Tür (6,87 m hoch und 3,27 m breit) führt in die kleine und fensterlose Cella, den Ort für die Aufstellung der Kultbilder. Dieser Raum wird heutzutage gelegentlich für Kunstausstellungen genutzt. Es sind keine Reste der antiken Inneneinrichtung erhalten.
[Wikipedia-Seitentitel: Maison Carrée
Datum der letzten Bearbeitung: 11. März 2023, 11:37 UTC
Versions-ID der Seite: 231706070
Permanentlink: https://de.wikipedia.org/w/index.php?title=Maison_Carr%C3%A9e&oldid=231706070
Datum des Abrufs: 6. Juni 2023, 21:22 UTC]

2 Alle Daten sind bis auf die beiden ersten Spalten fiktiv. Die italienischen Vor- und Nachnamen stammen großenteils aus dem Buch „Das Paradies der kleinen Sünder" von Andrea Camilleri (Imprint BLT der Verlagsgruppe Lübbe, Bergisch Gladbach 2001; Original: „Un mese con Montalbano", Arnoldo Mondadori Editore, Milano 1998). BLT hat der Nutzung dieser Namen zugestimmt.

Um jeglichem Konflikt aus dem Weg zu gehen, wurden die von Andrea Camilleri vergebenen Namen geändert, soweit sie Protagonisten betreffen, die in seinen Commissario-Montalbano-Krimis immer wieder vorkommen, oder es wurden den Nachnamen durch permutative Vertauschung andere Vornamen zugeordnet. Sogenannte „Allerweltsnamen" sind zwar nicht geschützt, aber es sollte damit vermieden werden, sich mit den Nennungen in der Tabelle direkt auf die Personen Camilleris zu beziehen.

3 [Wikipedia-Seitentitel: Lochkartensortierer
Datum der letzten Bearbeitung: 8. November 2021, 22:28 UTC
Versions-ID der Seite: 217112058
Permanentlink: https://de.wikipedia.org/w/index.php?title=Lochkartensortierer&oldid=217112058
Datum des Abrufs: 15. Juni 2022, 09:48 UTC]