Datenverknüpfung für die automatisierte Datenbereitstellung im Modellvorhaben

Thomas Berlage, Carlos Velasco, Fraunhofer FIT, 31.10.2025

Im Modellvorhaben müssen Daten bereitgestellt werden für Datendienste, sichere Verarbeitung, Qualitätssicherung durch den Plattformträger und für die Evaluierung. In vielen Fällen ist dabei eine Verknüpfung der Daten zwischen Klinischen Datenknoten und den Genom-Rechenzentren über die Vertrauensstelle erforderlich.

Hierfür ist ein Protokoll erforderlich, das damit umgehen kann, dass die Fälle eines Genom-Rechenzentrums über mehrere Klinische Datenknoten verteilt liegen können.

Im Folgenden wird ein solches Protokoll anhand eines Beispiels beschrieben. Das Beispiel geht davon aus, dass es 3 KDK ("KDK1", "KDK2", "KDK3") und 2 GRZ ("GRZ1", "GRZ2") gibt. Der gesamte Beispieldatensatz inklusive der jeweiligen Pseudonyme sieht wie folgt aus:

Vers.Nr.	KDK	kPseudo	Gender	GRZ	gPseudo	wgs
1001	KDK1	K11	m	GRZ1	G11	w
1002	KDK1	K12	f	GRZ1	G12	w
1003	KDK1	K13	m	GRZ1	G13	g
1004	KDK1	K14	f	GRZ1	G14	g
1005	KDK2	K21	m	GRZ1	G15	w
1006	KDK2	K22	f	GRZ2	G21	w
1007	KDK2	K23	m	GRZ2	G22	g
1008	KDK2	K24	f	GRZ2	G23	g
1009	KDK3	K31	m	GRZ2	G24	w
1010	KDK3	K32	f	GRZ2	G25	w

Der klinische Datensatz ist der Übersichtlichkeit halber nur durch "Gender" repräsentiert, der genomische Datensatz durch das Attribut "wgs", ob eine WGS-Sequenz ("w") oder ein Panel ("p") vorhanden ist. Die Pseudonyme (getrennt für KDK und GRZ) sind der Lesbarkeit und Nachvollziehbarkeit halber hier nicht komplett zufällig (und damit nicht korrekt).

Es gibt jetzt in der Realität keinen Knoten oder Akteur, der diese Gesamtsicht der Daten hat oder sie konstruieren könnte/dürfte. Die Aufteilung pseudonymisierter Daten auf die Knoten sieht wie folgt aus:

KDK1:

K11	m
K12	f
K13	m
K14	f

KDK2:

K21	m
K22	f
K23	m
K24	f

KDK3:

K31	m
K32	f

GRZ1:

G11	w
G12	w
G13	g
G14	g
G15	w

GRZ2:

G21	w
G22	g
G23	g
G24	w
G25	w

Die Vertrauensstelle kennt nur die Versichertennummer als Personenkennziffer sowie die zugeordneten verschiedenen Pseudonyme, die sie initial selbst vergeben hat (und die dazugehörigen Knoten-Ids).

Vers.Nr.	KDK	kPseudo	GRZ	gPseudo
1001	KDK1	K11	GRZ1	G11
1002	KDK1	K12	GRZ1	G12
1003	KDK1	K13	GRZ1	G13
1004	KDK1	K14	GRZ1	G14
1005	KDK2	K21	GRZ1	G15
1006	KDK2	K22	GRZ2	G21
1007	KDK2	K23	GRZ2	G22
1008	KDK2	K24	GRZ2	G23
1009	KDK3	K31	GRZ2	G24
1010	KDK3	K32	GRZ2	G25

Angenommen wird eine Abfrage, die alle Fälle sucht, die die Attribute Gender:"f" (aus den klinischen Daten aller KDK) und wgs:"w" (aus den genomischen Metadaten aller GRZ) haben. Die Query muss also zwischen den Knoten aufgeteilt werden, da es keinen Datenbestand gibt, an den ich die Frage zusammenhängend stellen kann. Der Einfachheit halber wird die anfragende Instanz hier "Broker" genannt, unabhängig davon, aus welcher technischen Umgebung die Anfrage herrührt.

Logisch gesehen ist die Query zwischen KDK-Daten und GRZ-Daten mit "UND" verknüpft. Wenn die Queries einfach verteilt werden, würden jeweils eine ganze Reihe von Datensätzen geliefert, die nach dem Join über eine gemeinsame Record-Id nicht mehr benötigt werden. Dies ist bei geringem Datenumfang tragbar, nicht aber wenn Attribute zum Beispiel komplex berechnet werden müssen (sei es die Berechnung eines Überlebenszeitraums aus zwei Daten oder die spezifische Prozessierung genomischer Rohdaten). Sinnvollerweise wird also erst der Join durchgeführt und die Daten erst im letzten Schritt abgerufen.

Die Anfrage beginnt also, indem der Broker eine Anfrage an alle KDKs schickt. Diese ist sinnvollerweise asynchron, das heißt, die Anfrage wird zunächst an alle verschickt, dann gewartet, bis alle geantwortet haben, um den Knoten die zeitlich parallele Berechnung zu ermöglichen.

Die Anfrage braucht eine zufällige Vorgangsnummer, um sich von anderen Anfragen zu unterscheiden. Die kann der Broker mit einem beliebigen Verfahren zur Generierung eindeutiger IDs selbst erzeugen, hier "Vorgang1".

Broker sendet an alle 3 KDK

Select *("Vorgang1", gender="f").*

Nach Ausführung der Selektion auf den internen Daten müssen die KDKs ihre eigenen Pseudonyme über die Vertrauensstelle in temporäre, dem Vorgang zugeordnete Pseudonyme transformieren, welche von der Vertrauensstelle für diesen Vorgang neu generiert werden. Gleichzeitig merkt sich die Vertrauensstelle diese Pseudonyme zusätzlich, um sie auch für andere Knoten weiterzuverwenden:

KDKs senden synchron an Vertrauensstelle
GetSecondaryPseudonyms *("Vorgang1", <Liste>) -> <Liste>:*
KDK1: <K12, K14> -> <v102, v104>
KDK2: <K22, K24> -> <v106, v108>
KDK3: <K32> -> <v110>

Die KDKs müssen das Ergebnis dieser Abfrage mindestens bis zur Datenlieferung zwischenspeichern, damit sie in der Lage sind, die sekundären Pseudonyme wieder in ihre eigenen rücktransformieren zu können.

Die Inhalte der Vertrauensstelle sehen jetzt so aus:

Vers.Nr.	KDK	kPseudo	GRZ	gPseudo	Vorgang1
1001	KDK1	k11	GRZ1	G11	
1002	KDK1	K12	GRZ1	G12	v102
1003	KDK1	K13	GRZ1	G13	
1004	KDK1	K14	GRZ1	G14	v104
1005	KDK2	K21	GRZ1	G15	
1006	KDK2	K22	GRZ2	G21	v106
1007	KDK2	K23	GRZ2	G22	
1008	KDK2	K24	GRZ2	G23	v108
1009	KDK3	K31	GRZ2	G24	
1010	KDK3	K32	GRZ2	G25	v110

Damit antworten die KDK jetzt an den Broker mit einer Pseudonymliste. Wir haben im Ergebnis 5 Fälle <v102, v104, v106, v108, v110> mit ihren klinischen Daten gesammelt, aber nur 3 davon sind WGS-Datensätze, was wir an dieser Stelle noch nicht wissen. Die klinischen Daten für die 5 Datensätze könnten jetzt übertragen werden, wir werden in diesem Beispiel aber erst am Ende Daten anfordern.

Der Broker muss weiterhin alle GRZ abfragen (das kann auch parallel geschehen):

Broker sendet an alle 2 GRZ
Select *("Vorgang1", wgs="w").*

Die GRZ führen jetzt ihre eigene Suche durch, dann werden auch diese Ergebnisse in die sekundären Pseudonyme rücktransformiert.

GRZs senden synchron an Vertrauensstelle
GetSecondaryPseudonyms *("Vorgang1", <Liste>) -> <Liste>:*
GRZ1: <G12> -> <v102>
GRZ2: <G21, G25> -> <V106, v110>

Die GRZs senden ihre Pseudonyme zurück, die im Broker zu <v102, v106, v110> zusammengeführt werden. Der Broker kann jetzt die Schnittmenge dieser Liste mit der Pseudonymliste der KDKs bilden.

Danach kann der Broker kann jetzt an allen Datenknoten die gewünschten Daten bezogen auf die in "Vorgang1" ausgewählten Pseudonyme abfragen:

Broker sendet an alle Knoten

GetData *("Vorgang1", <v102, v106, v110>, Attributliste)*

Die Knoten matchen die Anfrage mit den gespeicherten Ergebnissen von ***GetSecondaryPseudonyms***, sammeln die Attribute aus der spezifizierten Liste ein und kombinieren sie mit dem temporären Pseudonym.

Der Broker macht dann aus allen Meldungen durch Verknüpfung über das nun einheitliche Vorgangspseudonym eine Ergebnistabelle:

v102	f	w
v106	f	w
v110	f	w

Die Vertrauensstelle muss nur eine einzige Funktion implementieren:

GetSecondaryPseudonyms *(Vorgang, Dauer, <Pseudonym-Liste>) -> <Pseudonym-Liste>*

In dieser Funktion werden sekundäre Pseudonyme zugeordnet zu den primären Pseudonymen erzeugt. Diese Funktion muss/darf nur von den datenhaltenden Knoten aufgerufen werden. Die Pseudonyme dürfen nur von diesem Knoten stammen. Die sekundären Pseudonyme werden dem Vorgang zugeordnet und solange gespeichert, wie durch "Dauer" vorgegeben. Die Dauer wird vom Plattformträger festgelegt und kann von einem Tag (für temporäre Sessions) über "N Monate" (für Forschungsprojekte mit wiederholtem Zugriff) bis zu "M Jahre" reichen (für Abfragen, deren Ergebnismengen archiviert werden sollen).

Die Datenknoten müssen eine (asynchrone) Funktion implementieren

Select *(Vorgang, Dauer, Query)*

In dieser Funktion wird die angegebene Query auf den internen Daten ausgeführt. Für die Ergebnismenge werden sekundäre Pseudonyme angefordert. Der Datenknoten muss sich die sekundären Pseudonyme bzw. deren Zuordnung zu den internen Primärpseudonymen für die Dauer des Vorgangs merken, um später bei komplexen Joins nur für einen Teil der Ergebnismenge detaillierte oder berechnete Daten zu liefern. Alternativ könnte man die Zuordnung von der Vertrauensstelle anfordern, der diese auch bekannt ist (optionale Funktion ***GetPrimaryPseudonyms***). Dies ist allerdings etwas unsicherer, da die Gefahr besteht, dass bei Umgehen der Sicherungen Primärpseudonyme aus der Vertrauensstelle abgerufen werden könnten. Da die Vertrauensstelle eine zentrale Angriffsposition in der Dateninfrastruktur darstellt, ist es sinnvoll, deren Schnittstellen so klein wie möglich zu halten.

Ergebnisse der Selektion werden asynchron als Event geliefert, wenn die Prozessierung im Knoten abgeschlossen ist.

*Event **ResultSet** -> Vorgang, <Pseudonym-Liste>*

An dieser Stelle könnten auch gleich die Daten geliefert werden. Allerdings könnte ein großer Teil dieser Daten nach den folgenden JOIN-Operationen nicht mehr relevant sein. Da der aufrufende Broker auch eine vertrauenswürdige Instanz darstellt, ist dies in punkto Datenschutz nicht sehr bedeutsam, allerdings sollen die Knoten in Zukunft auch verteilt Berechnungen auf den lokalen Datensätzen ausführen können. In dem Fall ist es sinnvoll, die Berechnungen, die auch die Prozessierung genomischer Rohdaten umfassen könnten, auf die endgültige Ergebnismenge zu beschränken.

Daher sollte eine weitere asynchrone Funktion die Ergebnismenge beschränken können:

***SelectResult** (Vorgang, <Pseudonym-Liste>, ProcessingInstructions)*

Diese Funktion sucht aus der Pseudonymliste die eigenen Datensätze heraus (mittels der gespeicherten Zuordnung, andere Pseudonyme werden ignoriert) und ermittelt deren Primärpseudonyme. Darauf werden dann die *ProcessingInstructions* ausgeführt, die zunächst die gewünschten Attribute spezifizieren und gegebenenfalls auch Berechnungen anstoßen. Als Ergebnis wird dann ein zusammengestellter Datensatz geliefert (bzw. das Event signalisiert die Verfügbarkeit eines Ergebnisses zum Transfer). Die Art der *ProcessingInstructions* und das Protokoll des Datentransfers werden hier nicht weiter festgelegt.

*Event **Result** -> Vorgang, <Datensatz>*