Hi Kacper und ich haben hier ein bisschen was zusammengetragen:

1. Daten aus 1000 Genome Projekt:

Metadaten :

Geschlecht liegt vor
Der Lebens oder Geburtsort (Scheint Geburtsort zu sein)

Andere Daten sind nicht vorhanden oder wären unsinnig aufzunehmen (keine Krankheiten nur Verwandtschaft)

Metadaten sind in einem Gesonderten File zu finden und leicht zuordenbar.

Sequenzen: liegen hier als Bam File und hoffentlich als VCF vor.

Hier die Inhalte der beiden:
ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/technical/working/20120229_tutorial_docs/G1K_raw_data_and_ftp_20120217.pdf

BAM files

NAME	DESCRIPTION
QNAME	Query NAME of the read or read pair
FLAG	Bitwise FLAG (pairing, strand, mate strand etc
RNAME	Reference Sequence NAME
POS	1-Based leftmost POSition of clipped alignment
MAPQ	MAPpingQuality (Phred-scaled)
CIGAR	Extended CIGAR string (operations: MIDNSHP)
MRNM	Mate Reference NaMe('=' if same as RNAME)
MPOS	1-Based leftmost Mate POSition
ISIZE	Inferred Insert SIZE
SEQ	Query SEQuence on the same strand as the reference
QUAL	Query QUALity(ASCII-33=Phred base quality)

VCF Files

NAME	DESCRIPTION
CHROM	Chromosome name
POS	Position in chromosome
ID	Unique Identifer of variant
REF	Reference Allele
ALT	Alternative Allele
QUAL	Phred scaled quality value
FILTER	Site filter information

INFO User extensible annotation
FORMAT Describes the format of the subsequent fields, must always contain Genotype Individual Genotype Fields
These columns contain the individual genotype data for each individual in the file

wenn vorhanden werden wir aber nur die VCF benötigen.

Das Referenzgenom:
Das Referenzgenom sollte GRCh38 sein (ist internationaler Standard)

2.Ein paar Worte zum Entwurf:

Wir würden das ganze wie folgt lösen:

1. Jede Datenbank wird einzeln heruntergeladen (so wie es aussieht ist ein script oder java notwendig)
2. Für jede Datenbank erstellt ein Parser (lParser für lokal) ein InputFile (Das Format des Files wird weiter unten erläutert)
3. Ein weiterer Parser (gParser g für global)überträgt die Daten aus dem InputFile in die Datenbank. (Hierbei wird das Format des Files geprüft und wird dieses nicht eingehalten wird es nicht eingetragen.)

Mehr ist nicht notwendig. Das Ganze kann durch ein script oder eine java class gesteuert werden. Die InputFiles können z.b. 1,2,3,4.. usw. heißen und so leicht gefunden und vom gParser der Reihe nach abgearbeitet werden. Ähnlich wäre das mit den aufrufen der lparser und skripte möglich und somit wäre eine Erweiterbarkeit möglich und sehr einfach. (Nur Aufwand für den Kunden)

3.InputFile:
Hier das Format des Inputfiles:

//Kommentare sollten nicht eingefügt werden aber irgendwie muss man ja was erläutern

Referenzgenom:“Name des Referenzgenoms Bsp: GRCh38“
Referenzgenomsequenz:“wenn möglich hier die komplette Sequenz“
// Ansonsten erfolgt Aufteilung in mehrere Teile
// wenn es für männliche und weibliche Probanten unterschiedliche Referenzen gibt muss das
//kenntlich gemacht werden
$$
Quelle: „hier die Quelle angeben“
SampleID: „hier Samplename“
Chromosom/Genkoordinaten: „Angabe der Koordinaten“ // hier wird endweder Chromosom
//oder Genkoordinaten stehen und gemeint ist Koordinaten einer Mutation was ebend besser passt.
//Dies wird solange wiederholt bis keine neue Mutation vorkommt.
…..
…..

…..
//dann wieder $$
$$
SampleID: „hier SampleID aus der Datenbank“
//nun folgen Metadaten die noch nicht sicher sind
Gender: „m oder f“
Population: „drei Buchstaben bsp: GBR“
……

//Wenn eine Datenbank diese Info nicht enthält wird einfach # eingesetzt.
EOF

4. Die Datenbank:

Die Datenbank muss überarbeitet werden:

Referenz und Metadatenbank sind so okay nur das ReferenzID = Referenzname aus dem Inputfile und der Schlüssel für die Metadatentabelle also MetadatenID= Quelle+Sample.

Die Mutationstabelle müsste aber dann wie folgt aussehen

MutationsID=Quelle + Sample + Koordinaten // das ganze ist ein String
Quelle: String
AnfangMutation: long
EndeMutation: long
Chromosom: int
ReferenzID: String
Mutationssequenz: String
MetadatenID= Quelle+Sample // String

Somit kann schnell eine bestimmte Position/Intervall Gefunden werden.

Vorsicht!!! Die relative Häufigkeit ist hier noch nicht angegeben, weil nicht klar ist wie wir sie sinnvoll irgendwo einbinden können und die Häufigkeit ist aber gefordert. Hier ist also noch Arbeitsbedarf. (Gespräch mit den Projektleitern)

5.Testfälle:

Für die Tesfälle könnten nun falsche oder usinnige Inputfiles angegeben werden um den gParser zu testen. Da die lParser alle über lokalen Daten arbeiten könnten hier auch Beispielfiles angegeben werden und Inputfiles die die Ausgabe bilden sollen. Andere Testfälle wären nicht wichtig. (zumindest aus unserer Sicht)

Mehr Bleibt nicht zu sagen hoff ich :)