Bevor mit der Analyse einzelner Systeme begonnen werden kann, wird an dieser Stelle geklärt, welche Graph- und Aggregatsberechnungen eine geeignete Anwendung können soll. Hierzu werden die wichtigsten Anforderungen an ein System in der vorliegenden Arbeit im Folgenden spezifiziert:

Graphen werden in der vorliegenden Arbeit stets als Graph G = (V,E) betrachtet. Sie bestehen aus einer Menge V von Knoten und einer Menge E von Kanten, wobei eine Kante e immer zwei Knoten miteinander verbindet. Dennoch müssen noch weitere grundsätzliche Fragen der allgemeinen Graph-Struktur geklärt werden, bevor die Berechnung von Aggregaten detaillierter untersucht werden kann:

* Der betrachtete Teilgraph kann in seiner einfachsten Form aus einem einzelnen Knoten bestehen oder in der umfassendsten Analyse den gesamten Graphen darstellen. Alle Formen zwischen den beiden Extremen sind denkbar.
* Außerdem kann es nützlich sein, abgeleitete Formen des Graphen für sich zu betrachten. Als Beispiel hierzu dient eine Zusammenhangskomponente in einem ungerichteten Graphen. Ich definiere dann auch an den notwendigen Stellen kurz Begrifflichkeiten wie hier z.B. eine Zusammenhangskomponente. Richtig?
* Jeder Knoten besitzt **Properties**, die unterschiedliche Informationen und Werte des jeweiligen Knotens speichern. Sie können in Form eines Key-Value Paars dargestellt werden.

Was die zeitliche Komponente betrifft, kann bei einem Graphen zwischen einer statischen und einer dynamischen Sicht unterschieden werden:

* Bei einer statischen Betrachtung liegt der gegebene Graph von Beginn an in seiner jeweiligen Form vor und bleibt während des gesamten Untersuchungszeitraums unveränderlich.
* Im Gegensatz dazu betrachtet die vorliegende Arbeit besonders dynamische Graphen, die sich innerhalb des Betrachtungszeitraums beliebig ändern können. Es können daher sowohl Knoten als auch Kanten hinzukommen oder wegfallen. Außerdem sind auch die Properties der Knoten dynamisch, was bedeutet, dass sie neu hinzugefügt oder gelöscht werden können. Durch die Kombination einer Lösch-Operation und einer darauffolgenden neuen Einfüge-Operation kann so auch eine Update-Funktion realisiert werden.
* Der Kern temporaler Graphen bezieht sich darauf, dass sowohl Knoten als auch Kanten zu verschiedenen Zeitpunkten oder Zeitintervallen aktiviert sind oder nicht. Entweder sind die Kanten mit daher mit einem oder mehreren Timestamps versehen, die angeben zu welchem Zeitpunkten die Verbindung zweier Knoten aktiv besteht, oder die Kanten sind mit Zeitintervallen annotiert. Z.B. besteht zwischen Knoten A und B eine Kante mit Intervall [01/18 – 04/18], woraus ersichtlich wird, dass zwischen Januar 2018 und März 2018 die Knoten A und B in Verbindung zu einander standen.
* In diesem Zusammenhang wichtig ist folgende getroffene Voraussetzung: zwei Intervalle einer Kante dürfen sich nicht überlappen.

Nachdem festgelegt wurde, wie die generelle Struktur eines betrachteten Graphen aussieht, wird als nächstes der Begriff der Aggregatsberechnung innerhalb der Ausarbeitung definiert. Es wird festgelegt, welche Formen von Aggregaten berechnet werden sollen:

Aggregate: auf Menge von Knoten laufen lassen. Menge oder Liste von Eingaben, als Ausgabe ein Wert. Graph als Knotenmenge und die aggregieren – ist das die Lösung? Inkrementelle Aggregate auf Mengen, nachdem Mengen effizient verwaltet. Jede Menge eigner Strom, der sich verändert. Mengen können hierarschisch sein. Naiad Strom und Iteration. Ein DAG von Mengen. Embgen und Bierbrauer Arbeiten.

* Zum einen soll es möglich sein, Aggregate an einem einzelnen Knoten oder für den Gesamt-Graph zu berechnen.
* Auch an den Kanten?
* Es wird angenommen, dass die Aggregate durch die Werte der Properties an den Knoten berechnet werden.
* Das hieraus folgende Ziel ist, dass sich die aggregierten Werte somit dynamisch ändern, sobald sich der Graph und die Properties verändern.

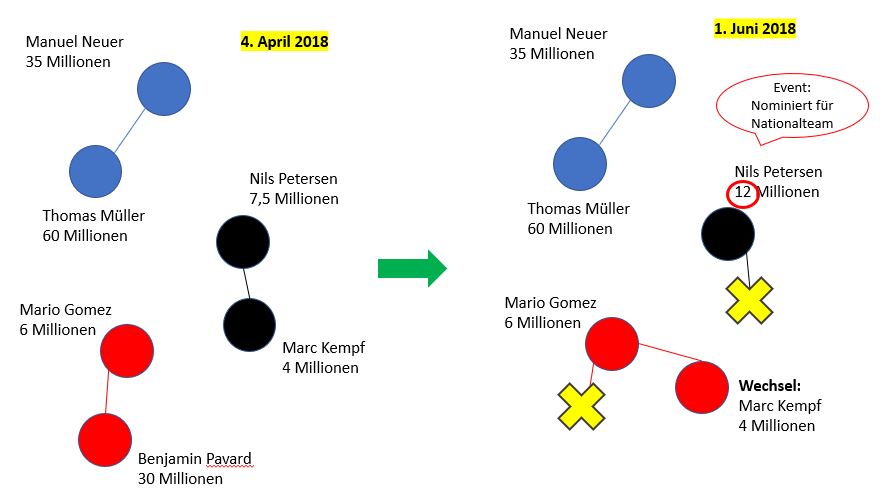
Im nächsten Abschnitt werden nun ausgewählte Systeme vorgestellt, die auf die hier beschriebenen Anforderungen untersucht werden. Das Ziel der Ausarbeitung ist es eine Anwendung, inkl. des Open-Source Codes, zu finden, die dann den folgenden Use Case eines dynamischen Graphen verarbeiten kann.

Beispiel Use-Case: Bundesliga Graph

* Jeder der 18 Bundesliga Vereine hat ca. 25 Spieler im Kader auf www.transfermarkt.de
* Ein Verein stellt quasi eine Zusammenhangskomponente dar
* Wechselt ein Spieler innerhalb der Bundesliga, dann fallen Kanten weg und neue entstehen
* Verlässt ein Spieler die Bundesliga oder beendet er seine Karriere, dann fällt der Knoten weg
* Ca. alle 6 Monate Wechselzeit und neue Spieler kommen hinzu oder verlassen den Bundesliga Graph
* Jeder Spieler hat einen Marktwert, der sich wöchentlich ändern kann, realistisch eher monatlich
* Denkbare Aggregate:
  + MAX: Welcher Spieler im Verein hat den höchsten Marktwert? Welcher Spieler ist der jüngste Spieler der Bundesliga?
  + SUM/MAX: Welcher Verein hat den größten Kader (Spieleranzahl)? Welcher Verein weißt den höchsten Gesamt-Marktwert aller Spieler auf? Welche Nationalität ist wie oft in der Bundesliga vertreten?
  + AVG/MIN: Welche Mannschaft hat das geringste Mindestalter der Spieler?
  + Nur an einem Knoten: über die Zeit veränderter Marktwert - wann und was war der maximale Marktwert eines Spielers?
* Dynamischer Graph mit einer Vielzahl an möglichen Properties an den Knoten. Für den weltweiten Fußball denkbar, wodurch auch eine große Menge an Daten entstehen würde.

Allgemeine Probleme der aktuell untersuchten Systeme: Stand-alone Systeme, kein einheitlicher Standard, was es schwierig macht den Code wiederzuverwenden und die Systeme miteinander zu vergleichen. (Die meisten fokussieren sich auf Timestamps und nicht Intervallen?) Daher scheint zum jetzigen Stand der Forschung nötig, z.B. auf Flink zurückzugreifen und mit Built-On-Top Systemen den Use zu realisieren.

Alle „alten“ Systeme (Pregel und Co.) haben mit den gleichen 3 Hürden zu kämpfen. **Speichern** des Graphen (Snapshot-Format z.B.), **Berechnung** und **Kommunikation**. Dann neue Ansätze: …



Das Kapitel schließt mit folgender Arbeitshypothese: Die betrachteten Systeme bieten eine Vielzahl an Ideen, wie temporale Graphen verarbeitet werden können, allerdings ist kein System so ausgereift, dass es einen entsprechenden Quellcode oder eine API zur Verfügung stellt, die den beschriebenen Use Case in vollem Umfang abbilden kann.

Pregel, GraphLab etc. -> Kineograph, Chronos (Immmortalgraph), Tornado, Naiad, GraphInc -> on top of Spark bzw. Flink Tegra und Tink. Sonst GraphStream (kein Fokus auf temporale Graphen, aber scheint verwendbar und liegt Open-Source vor).

Mögliche Grob-Gliederung:

1. Ziel/Problemdefinition der Arbeit und Anforderung an Systeme und Use Case
2. Pregel und die Beginne von Graph Processing
3. Systeme wie Kineograph und Tornado inkl. Gesamtübersicht in Matrix
   1. Ansätze, nicht exakt das, was wir suchen
4. Tink (Flink) und Tegra (Spark) Ansätze
5. Versuch den Use Case umzusetzen – a) was eigenes oder b) was on top of Flink z.B. oder c) in GraphStream

**Wie modellieren die einzelnen Systeme temporale Graphen?**

**Pregel**

* Temporales Modell: nicht vorhanden
* Graph-Algorithmen: jeder Knoten kommuniziert mit Nachbarn und tauscht Nachrichten aus, Informationsverarbeitung führt zu möglichen Änderungen in Superstep S + 1.
* Aggregate: jeder Knoten kann Werte an einen Aggregator senden (in Superstep S), der Aggregator verarbeitet die Werte durch Reduktion und teilt allen Knoten in Superstep S+1 den neuen Aggregats-Wert mit.

**GraphLab**

* Ausgeschlossen, weil zu weit weg vom Thema Temporale Graphen

**PowerGraph**

* Verweist selbst darauf, dass eine Erweiterung für temporale Graphen noch fehlt
* Fasst GraphLab und Pregel gut zusammen

**GraphChi**

* Große real-world Graphen nicht durch verteilte Bearbeitung, sondern als disk-based system mit der Technologie **parallel sliding window**
* Über die Zeit sich weiterentwickelnde Graphen werden unterstützt
* Besonderer Fokus auf dem Hinzufügen von Kanten durch das Implementieren eines vereinfachten I/O Buffer Baums.

**Kineograph**

* Stream von Daten als Eingabe, um einen sich stets verändernden Graphen zu erstellen
* Serie von Snapshots durch EPOCH COMMIT PROTOCOL
* Temporales Modell: Ingest Nodes erhalten neue Daten, analysieren die Daten und geben sie an Graph Nodes weiter
* Snapshot Mechanismus: Graph Nodes speichern die Graph Updates der Ingest Nodes, danach Ingest Nodes melden Graph Update in einer zentralen und globalen PROGRESS TABLE. Snapshooter instruiert alle Graph Nodes einen Snapshot zu machen je nach aktueller Sequenznummer. Abschließend folgt inkrementelle Berechnung.

**Chronos**

* In-memory, iterative Berechnung auf temporalen Graphen
* Temporal graph mining der hier verwendete Begriff für eine Serie von Snapshots

**Chronograph**

**TGraph**