Inwiefern Batch oder Stream an welcher Stelle der Verarbeitung sinnvoll/performant sind, ist Teil dieser Masterarbeit, daher sollten alle Ideen gesammelt und in sinnvoller Reihenfolge ausprobiert werden, hier erstmal ungeordnet:

* Wie oben formuliert ist eine Batchverarbeitung nur dann sinnvoll, wenn für die jeweilige Operation das ganze Datensetbetrachtet werden muss ODER das Datenset ungeordnet vorliegt. Da das nicht für gegeben angenommen werden muss, wird vorerst der Streaming-Ansatz verfolgt
* **Graph-Daten als Stream: Die Graphdaten werden als Stream dargestellt und die Datenquelle ist optimalerweise bereits geordnet. Hier gibt es verschiedene Ansätze/Überlegungen:**
  + **Die Knoten der Top-Ansicht sollten vorne im Stream liegen, um nicht den ganzen Stream einlesen zu müssen. Mit einem Count-based Window kann auf einer vorsortierten Knotenmenge eine von den Viewport-Constraints definierte Anzahl an Knoten ausgewählt und visualisiert werden.**
  + **Man unterscheidet zwischen Zoom- und Panning-Operationen. Der Panning-Operation (Bewegung in gleicher Vergößerungsebene) sollte mehr Bedeutung zugeordnet werden, weil eine Verzögerung hier die Interaktivität stärker behindert als in der Zoom-Operation. Dafür könnte ein Vorladen der 8 Nachbarquadranten der jeweiligen Zoom-Ebene hilfreich sein. Da unbekannt ist, in welchen Quadranten der Anwender als Erstes zoomen wird, ist auch unbekannt, welche die Nachbarquadranten sein werden und können daher nicht im Vorhinein in die Ordnung des Streams integriert werden.**
  + Viewport-Constraints:
    - **Viewport definiert die Zahl der darzustellenden Knoten**
    - Je nach Anzahl der Knoten im Graphen kann man eine unterschiedliche Anzahl an Zoom-Ebenen definieren und daraus die Reihenfolge der Knoten im Stream konstruieren. Nachbarknoten der Knoten der Top-Ansicht sollten vorrangig visualisiert werden. Da allerdings unbekannt ist, welche Anzahl an Knoten der Viewport in der Top-Ansicht abbilden wird, ist auch unbekannt welches die Nachbarknoten sein werden. In der Annahme das immer eine definierte Anzahl an Knoten in der Top-Ansicht dargestellt wird, wäre diese Ordnung möglich, allerdings ist die Darstellung dann weniger flexibel gegenüber verschiedenen Viewports.
  + **Flink-intern gilt es herauszufinden, welche Datenstrukturen und Operationen performant sind. Folgende Ideen dazu:**
    - **Ungeordnet, Top-Ansicht: Darstellung der Knoten des Graphen u.a. als Degree-Matrix (oder andere Metrik)auf der nach einer bestimmten Anzahl Knoten mit definiertem Grad-Bereich abgefragt werden kann.**
    - Geordnet, Top-Ansicht: Darstellung der Knoten als geordnete Objektfolge
      * Flink Table: Nach Knotengrad (oder anderer Metrik) sortierte Tabelle. SQL-Anfragen möglich.
      * Flink Streaming: In Verbindung mit bspw. Apache Kafka (Knoten als geordnete Events in einem Topic) ein geordneter Stream von Knotenobjekten.
    - **Ungeordnet, Zoom-Ebene mit Nachbarknoten:** 
      * **Flink Table: Adjazenzmatrix**
      * Flink Streaming: Verkettung von Knotenobjekten (im Prinzip EPGM-Modell), realisierbar z.B. über Verweis auf anderes Topic
    - Geordnet, Zoom-Ebene mit Nachbarknoten:
      * Flink Table: Adjazenzmatrix, in welcher die Knoteneinträge nach gegebener Metrik geordnet sind.
      * Flink Streaming: Einträge der Adjazenzmatrix als geordnetes Topic nach gegebener Metrik, Stream kann ab dem letzten Knoten der Top-Ansicht eingelesen werden.
  + Super-Knoten und Edge-Bundling. Wenn das Graph-Layout diese Eigenschaften haben soll, ist eine vorheriges Erstellen des Layouts notwendig, da diese Elemente nicht ohne Betrachtung des gesamten Datensets erstellt werden können. Eine Betrachtung des gesamten Datensets in der Visualisierung ist nicht performant.
  + **Layout vor der Visualisierung gegeben.**
    - **Streaming-Windows befüllen, Ideen:**
      * **Jeweils Einteilung der Zoom-Ebenen in Quadranten, mit welchen die Windows befüllt werden (der Viewport muss dann nicht mehr zwangsweise ausgeschöpft werden). Bei einer Zoom- oder Panning-Operation werden dann die entsprechenden Windows getriggert. Wenn ein Quadrant der Größe des Viewports entspricht, dann werden maximal 4 Windows neu getriggert, bei horizontalem/vertikalem Panning nur zwei.**
  + Layout ad hoc bei der Visualisierung (Super-Knoten und Edge-Bundling nicht performant, siehe oben):
    - Mehr Raum für Knoten mit hohem Grad.
    - Zur Übersichtlichkeit sollte die Platzierung von Knoten im Layout nicht mehr verändert werden, sobald sie das erste Mal visualisiert wurden. Dadurch wird das Layout nicht optimal, die Erstellung jedoch performanter.
    - Streaming-Windows befüllen, Ideen:
      * Für jeden Knoten der Top-Ansicht wird ein Window mit Nachbarknoten befüllt. Bei Zoom in die erste Unterebene werden alle Windows mit Knoten aus der Top-Ansicht getriggert, welche noch im Viewport enthalten sind. Der Viewport bestimmt die Anzahl der zu visualisierenden Knoten. Wenn Knoten mehrmals getriggert werden, werden diese bevorzugt visualisiert (Stream-Reihenfolge).