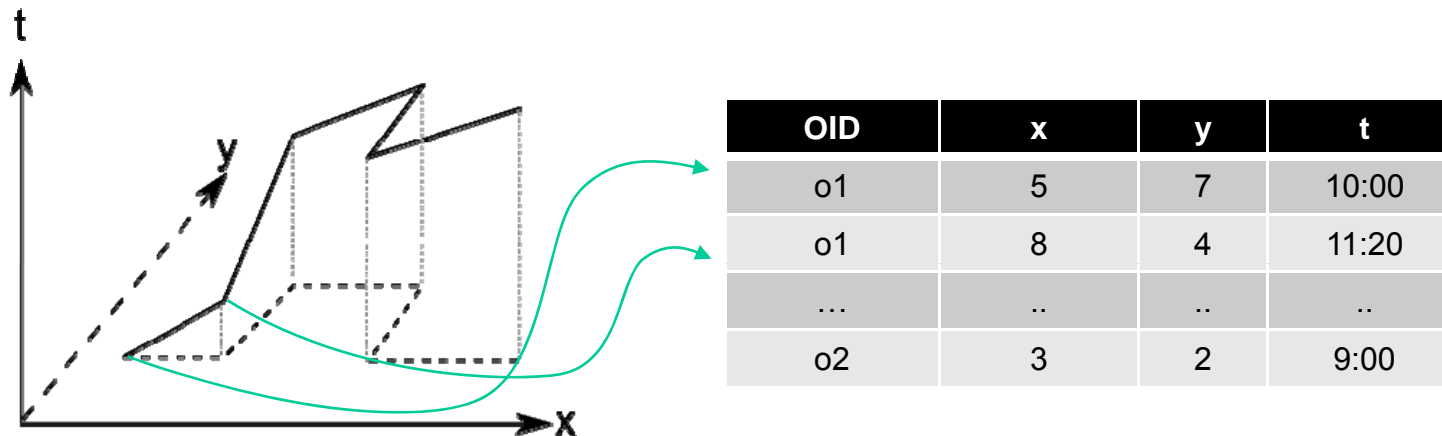


3.2.2 Anfragen an die Vergangenheit

– Darstellung von Historischen ST-Daten

- Abbildung der (vergangenen) Bewegung eines Objektes über Trajektorien (Sequenz von Positionen (Beobachtungen); Verlauf zwischen Beobachtungen oft linear interpoliert)
- Zeit-Attribut wird oft als zusätzliche Dimension betrachtet:
 - Objekte die sich im d -dimensionalen Raum bewegen werden in einen (statischen) $(d+1)$ -dimensionalen Raum transformiert.

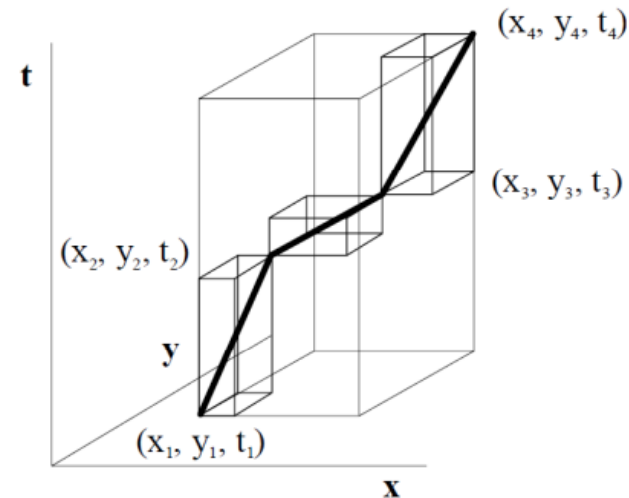
=> Trajektorien als Sequenz von $(d+1)$ -dimensionalen Punkten



– Verwaltung der Vergangenheit

- Idee:

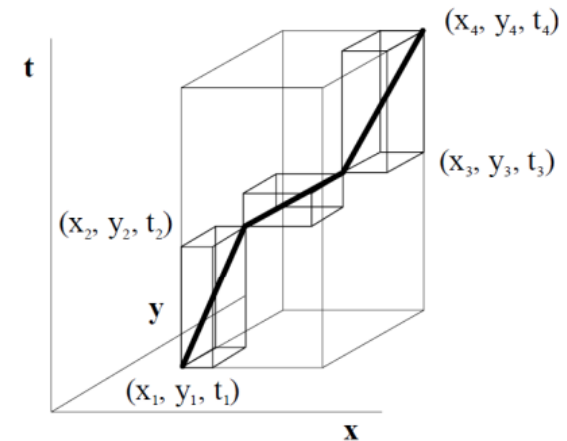
- Erweiterung des R-Baums um eine weitere Dimension, die Zeit t
- Die Trajektorie eines Objektes kann als Sequenz von $(d+1)$ -dimensionalen Liniensegmenten repräsentiert werden
- Die einzelnen Liniensegmente werden in einem R-Baum abgelegt



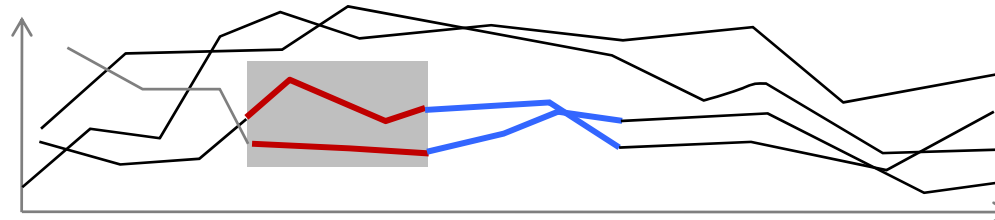
- Beispiele: 3D-R-Tree [MGA03], STR-Tree und TB-tree [PJT00]
- ST-Index-Varianten unterscheiden sich gemäß der Gruppierung der Objekte auf die Index-Seiten (unterschiedliche Splitstrategien bei Überlauf von Seiten)

– Der 3D-R-Tree: [TVS96]

- Prinzip:
 - Verwaltung der $(d+1)$ -dimensionalen Liniensegmente in einem herkömmlichen R-Baum (R^* -Baum).
- Vorteile:
 - Verwendung einer bereits existierenden Indexstruktur
 - Unterstützt alle Snapshot-Anfragen (d.h. räumliche Anfragen zu einem Zeitpunkt oder über einen Zeitbereich)
 - Einfach zu implementieren
- Nachteile:
 - Segmente der Trajektorie eines Objektes werden als „unabhängige“ Objekte/Einträge betrachtet
 - => Eine Trajektorie wird (willkürlich) über mehrere Seiten verstreut
 - => Problem für ST-Anfragen über einen größeren Zeitbereich
 - Seitenregion wachsen zu schnell in der Zeitachse → zu wenig Information für einen Zeitpunkt
 - => Problem bei Anfragen die sich auf einen Zeitpunkt beziehen

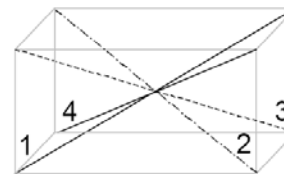
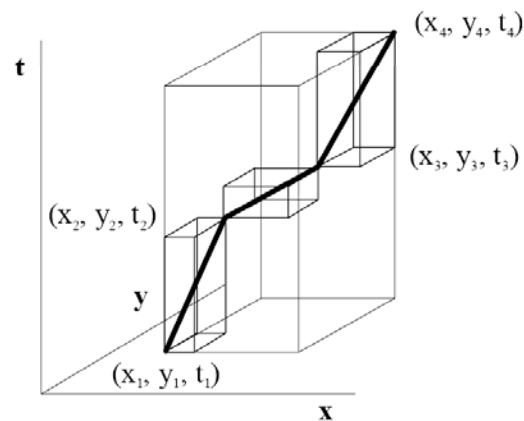


- Weiterer Nachteil:
 - Oft werden zusammenhängende Trajektorien-Segmente zusammen angefragt, wie z.B. bei **Trajektorien-basierten ST-Anfragen**.
 - Bei Trajektorien-basierten Anfragen unterscheidet man zwischen:
 - » Teiltrajektorie die für die Erfüllung des Anfrageprädikates relevant ist.
 - » Teiltrajektorie die als Anfrageergebnis ausgegeben wird.
 - Beispiel:
 - » Wo haben sich die Personen die sich **zwischen 8 und 9 Uhr auf dem Kirchplatz** aufgehalten haben **innerhalb der folgenden Stunde** hinbewegt?



- Anfrage wird in 2 Schritten durchgeführt:
 - » 1) **Selektion der Trajektorien** über ST-Anfrage: „...zwischen 8 und 9 Uhr auf dem Kirchplatz ...“
 - » 2) **Ermittlung der Ergebnis-Trajektorien** die auszugeben sind: „...innerhalb der folgenden Stunde hinbewegt.“
- Zugriff auf Zusammenhangskomponenten werden im 3D-R-Tree nicht effizient unterstützt.

- Der Spatio-Temporal-R-Baum (STR-Baum) [PJT00]
 - Struktur analog zum 3D-R-Baum:
 - Erweiterung des R-Baums um die Zeit-Dimension.
 - Unterschied zum 3D-R-Baum:
 - Einfüge und Split-Heuristik (siehe später)
 - Zusätzlich: Auf Blattebene ist zu jedem MUR-Eintrag ein Schlüssel für die Orientierung von dem jeweilig approximierten Trajektorien-Segment abgespeichert:
=> Trajektorien-Segment kann mit Hilfe des Schlüssels direkt aus den MUR-Daten ermittelt werden.

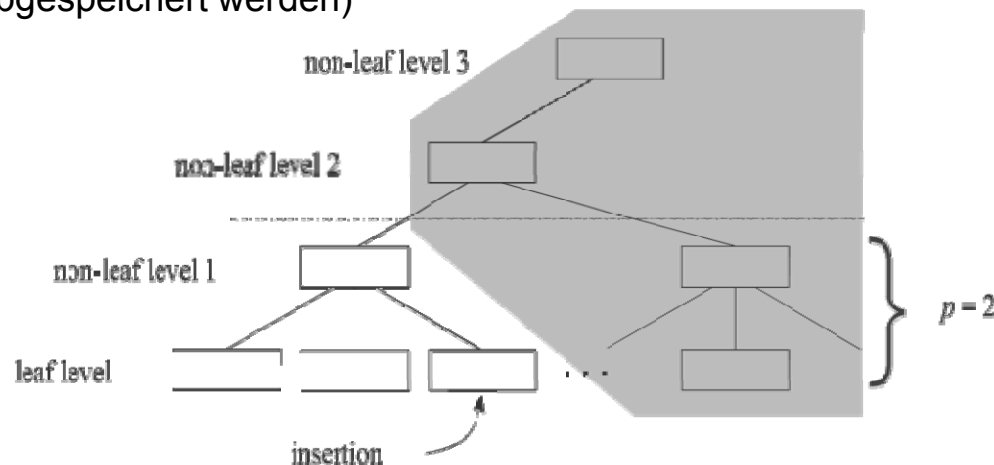


In einem (2+1)-dimensionalen MUR gibt es für ein Segment nur 4 unterschiedliche Orientierungen.

- Auf Blatt-Ebene werden Einträge der folgenden Form abgespeichert:
(id,tid,MUR,orientation)
id=(Objekt/Trajektorien)-ID,
tid=Nummer des Segments in der Trajektorie,
MUR=minimal-umgebendes Rechteck des Segments,
orientation=Orientierung des Segments innerhalb des MURs.
 - Ziele:
 - Speichere (topologisch nahe) zusammenhängende Trajektoriensegmente möglichst auf der gleichen Datenseite ab:
 - » Unterstützung von Anfragen über längere Zeiträume.
 - » Unterstützung von Trajektorien-basierten ST-Anfragen.
 - Gruppierung von räumlich nahen Segmenten (unterschiedlicher Trajektorien) soll nicht vernachlässigt werden:
 - » Anfragen über kurze Zeiträume (Time-Slice Queries).
- => Grundlegende Strategie: Trade-off zwischen beiden oben genannten Zielrichtungen.

- Einfüge-Strategie

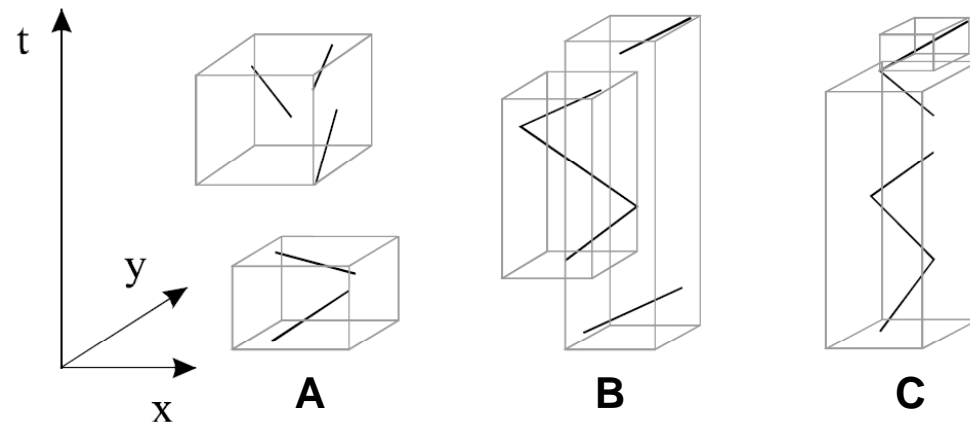
- Idee: Objekte werden nicht nur aufgrund räumlicher Nähe, sondern anhand ihrer Zugehörigkeit zu einer Trajektorie in den R-Baum eingefügt
- Einfügen eines Trajektorien-Segments:
 - » Suche den Knoten mit dem Vorgängersegment des einzufügenden Segmentes
 - » Knoten ist nicht voll: -> Füge neues Segment in den Knoten ein
 - » Knoten ist bereits voll:
 - Ist einer der Eltern des Knotens in den $p-1$ darüber liegenden Hierarchieebenen nicht voll, dann splitte das Blatt (Parameter p kann frei gewählt werden und bestimmt in welchem Nachbarschaftsradius Segmente der gleichen Trajektorie abgespeichert werden)



- » Falls $p-1$ Elternknoten voll, füge das Segment anhand der R-Baum-Einfüge-Strategie in einen andern $p-1$ -Teilbaum ein (grau-schraffierte Zone).

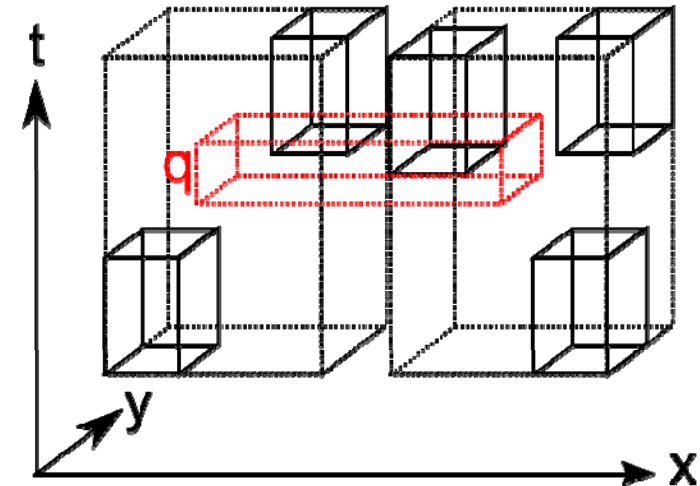
- Split-Strategie

- Idee: Füge immer die jüngsten Segmente in den neuen Knoten ein, denn für diesen ist es am wahrscheinlichsten, dass dort künftige Nachfolgesegmente (der gleichen Trajektorie) eingefügt werden.
- Blattknoten:
 - » Alle Segmente im Blatt sind nicht miteinander verbunden (**A**): Verwende Quadratischen Algorithmus des R-Baumes
 - » Es gibt mindestens ein nicht-verbundenes Segment (**B**): Nicht verbundene Segmente werden in den neuen Knoten eingefügt
 - » Es gibt kein nicht-verbundenes Segment (**C**): Suche das bzgl. der Zeitachse „jüngste“ Segment und füge es in den neuen Knoten ein.



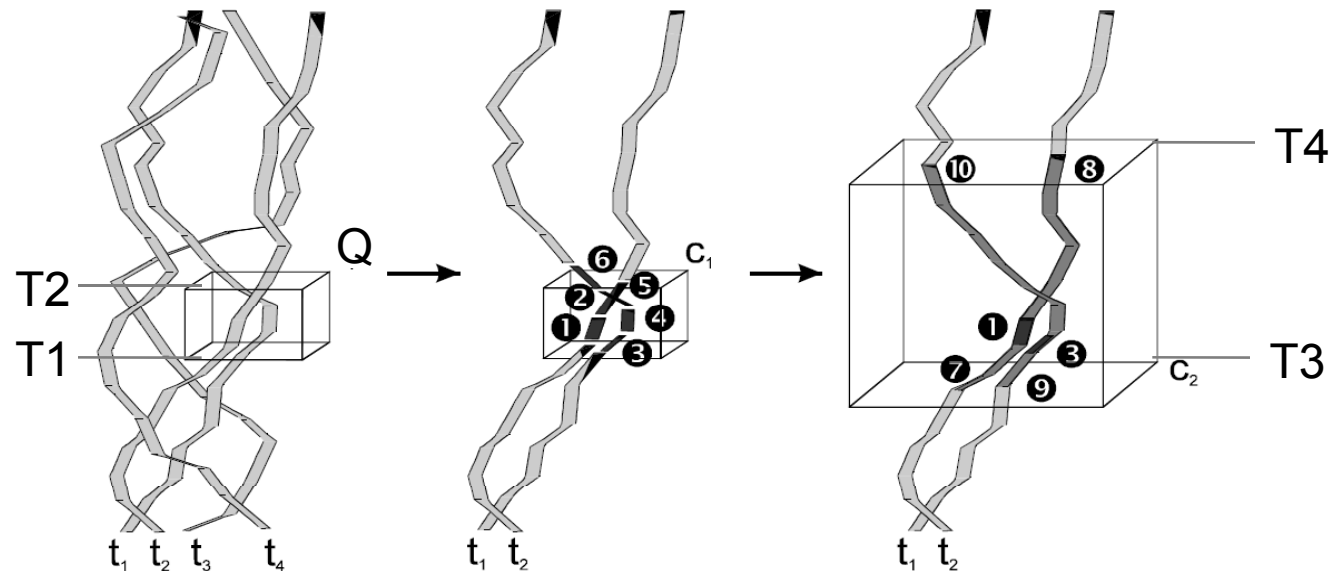
- Nicht-Blatt Knoten:
 - » Erstelle neuen Eintrag für den neuen Knoten (alter Knoten bleibt unverändert)

- ST-Fenster-Anfrage auf den STR-Baum
 - Anfrage „Liefere mir alle Objekte, die sich innerhalb des Zeitraums (t_1, t_2) im Fenster (x_1, y_1, x_2, y_2) befanden.“
 - Prinzipiell wie eine Fensteranfrage auf den R-Baum
 - Einstieg bei der Wurzel und rekursiver Abstieg bis zur Blattebene
 - Anfrage liefert alle Segmente/Trajektorienabschnitte, die das Anfragefenster schneiden.
 - Alternativ können auch nur die Objekt-IDs der resultierenden Objekte zurückgegeben werden.



- Trajektorien-basierte ST-Fenster-Anfrage auf dem STR-Baum
 - Finde zu allen Objekten die sich zwischen $T1$ und $T2$ im Abschnitt Q befunden haben die Vorgänger- und Nachfolge-Trajektorie im Zeitintervall $T3$ und $T4$:

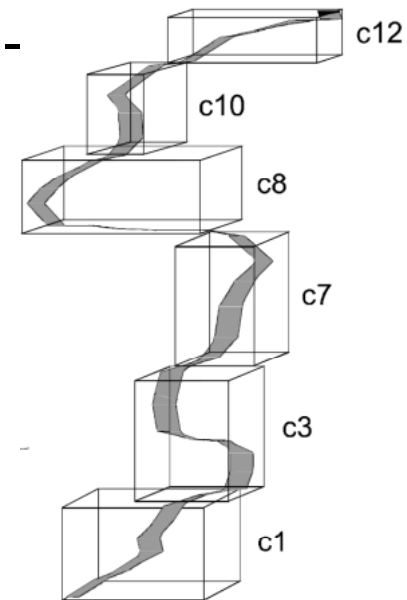
- Die Trajektorien-basierte ST Fenster-Anfrage wird wieder im 2 Schritt-Modus Ausgeführt:



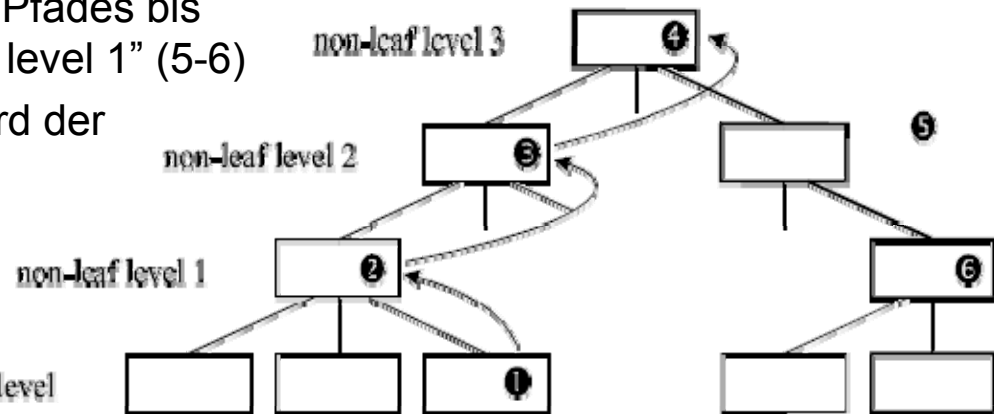
- Schritt 1) **Selektion der Trajektorien**: Fensteranfrage auf den STR-Baum mit Anfragefenster Q
- Schritt 2) **Ermittlung der Ergebnis-Trajektorien**: Verfolgung der in 1) ermittelten Trajektorienabschnitte durch rekursive Bereichsanfrage
- Endpunkt des aktuellen Segments wird als Anfrageobjekt verwendet
- zunächst wird innerhalb der gleichen Seite nach den angrenzenden Segmenten gesucht
- Falls die aktuelle Seite keine direkt angrenzenden Segmente enthält wird die Suche als neue Bereichsanfrage (auf den vollen Index) gestartet.

– Variante: TB-Baum [PJT00]

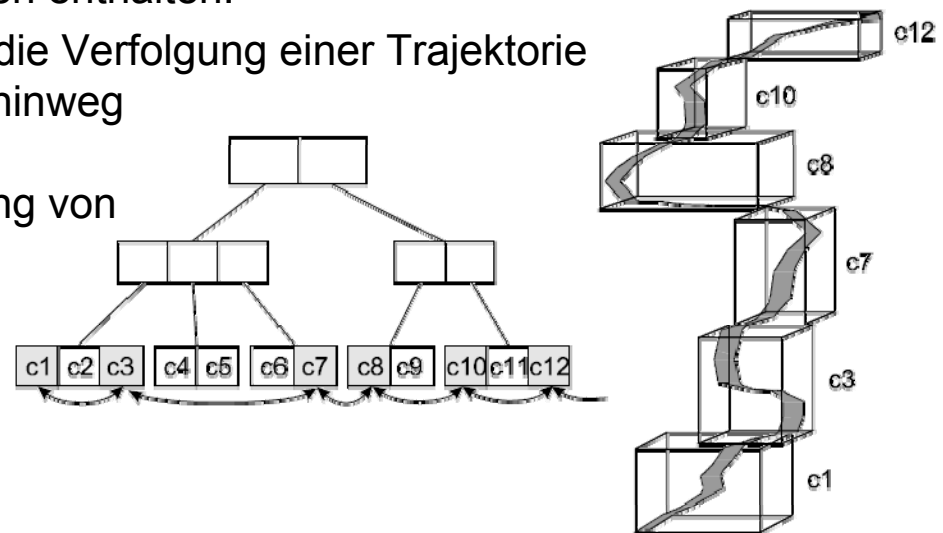
- Jedes Blatt enthält **nur Segmente**, die zur **gleichen Trajektorie** gehören
 - **Ziel:** Stärkere Unterstützung von Trajektorien-basierten Anfragen
 - Neue Segmente werden immer in den Blattknoten eingefügt, der das letzte Segment der Trajektorie enthält
 - Angepasste Insert- und Splitstrategien
 - **Nachteil:** Nahe Segmente, die zu unterschiedlichen Trajektorien gehören, werden in unterschiedlichen Knoten abgelegt
- => Schlechte Unterstützung von Bereichsanfragen die sich auf nur einen Zeitpunkt beziehen (Time-Slice Queries).



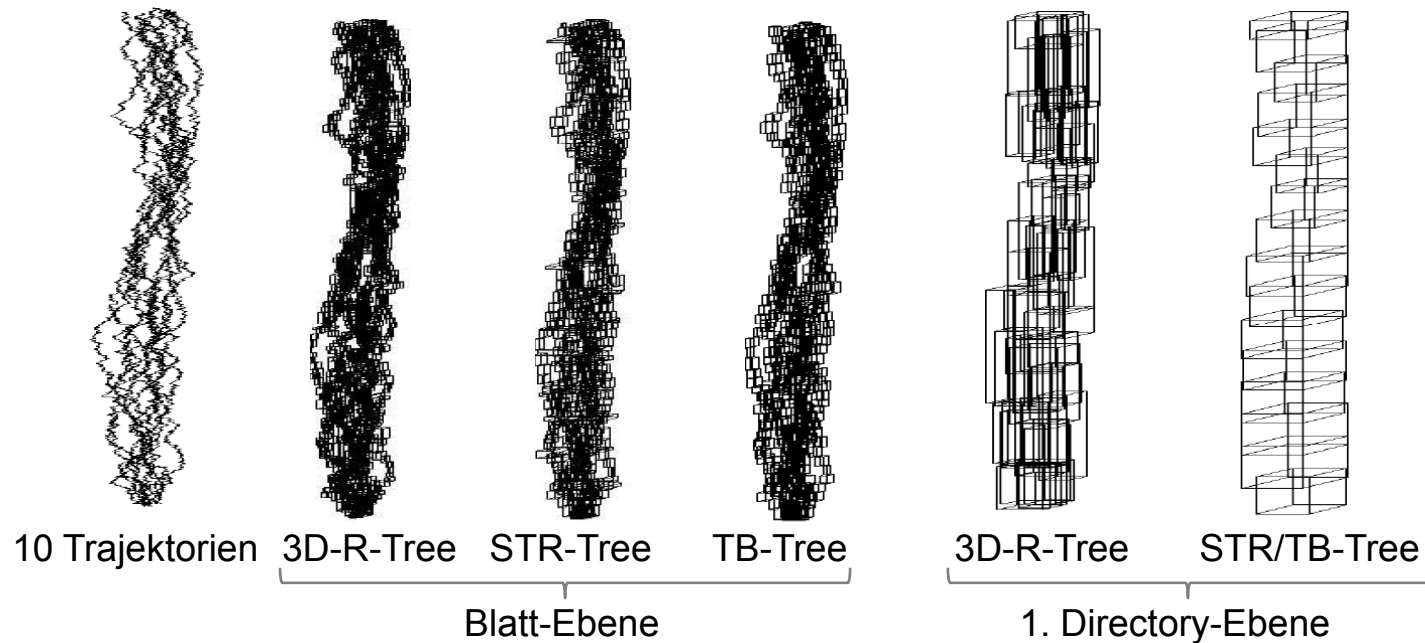
- Einfüge-Strategie:
 - Einfügen eines neuen Segments:
 - 1) Suche nach den Blattknoten mit dem Vorgängersegment
Bereichsanfrage mit dem MUR des einzufügenden Segments als Anfrageobjekt.
Rekursiver Abstieg des TB-Baums
 - 2) Falls der Blattknoten nicht voll ist wird das neue Segment eingefügt.
Ansonsten wird eine **Splitstrategie** initiiert.
- Split-Strategie:
 - Erzeugung eines neuen Blatt-Knotens N
 - Suche nach Vaterknoten der nicht voll ist (rekursiver Aufstieg) (2 – 4)
 - Abstieg des rechtesten Pfades bis Knoten N' auf "non-leaf level 1" (5-6)
 - Falls N' nicht voll ist, wird der Knoten N angehängt
 - Ansonsten (N' voll) wird N' gesplittet und ein neuer Teilpfad mit dem Blattknoten N erstellt.



- Eigenschaften des TB-Baum:
 - Blattebene enthält nur Knoten mit jeweils einer Teiltrajektorie.
 - Eine Trajektorie wird somit über eine Menge von Blattknoten organisiert.
 - Segment-ID muß nicht für jedes Segment abgelegt werden (vgl. STR-Baum), sondern kann als gemeinsame Segment-ID im jeweiligen Blattknoten mitabgespeichert werden.
 - TB-Baum wächst von links nach rechts (Einfügen neuer Knoten im rechten Teilbaum; siehe Splitstrategie)
 - Erhaltung der Trajektorie über zusätzliche Querverlinkung von Blattknoten: Zwei Blattknoten sind bi-direktional miteinander verlinkt wenn sie angrenzende Teiltrajektorien enthalten.
 - Durch die Verlinkung wird die Verfolgung einer Trajektorie über mehrere Blattknoten hinweg sehr effizient unterstützt.
(Vgl. STR-Baum: Verfolgung von Trajektorien über Bereichsanfragen)

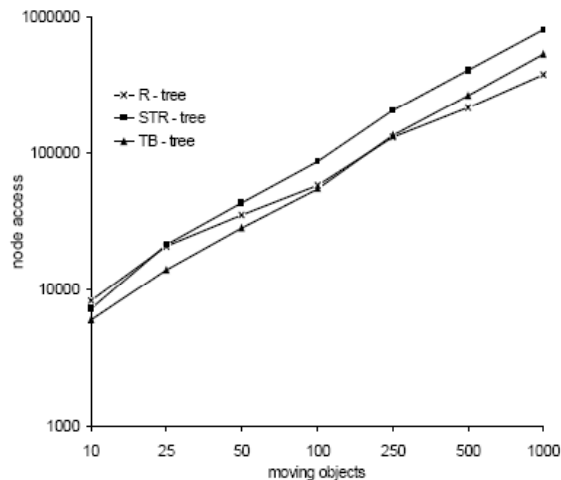


- Vergleich der Indexstrukturen
(3D-R-tree, STR-tree, TB-tree)
 - Überlappung der Seitenregionen:

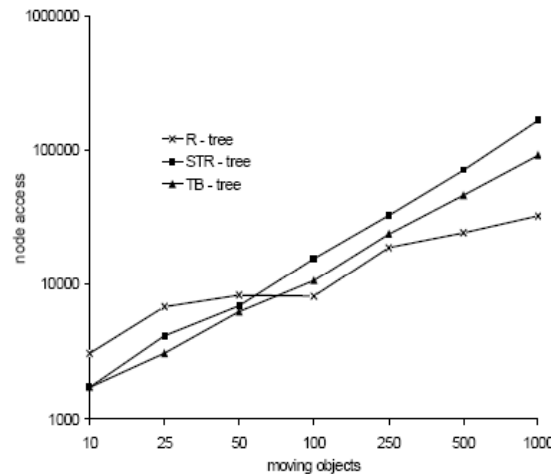


- Auf Blattebene sind keine großen Unterschiede zwischen allen 3 Indexvarianten zu erkennen.
- Auf der 1. Directory-Ebene deutlich größere Überlappung der Seitenregionen beim 3D-R-Baum als beim STR-Baum und TB-Baum.

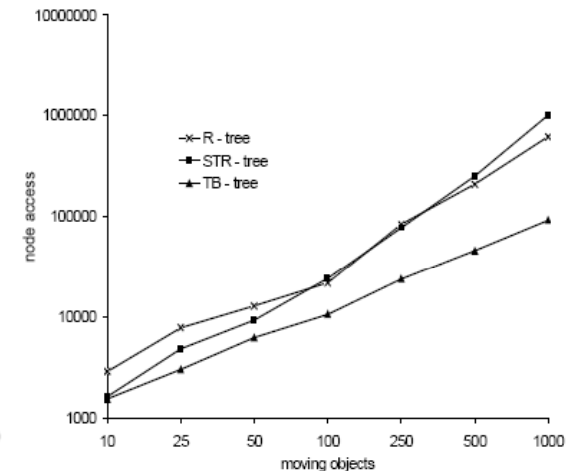
- Anfrageperformanz:



Bereichsanfragen
(über Zeitbereich)



Bereichsanfragen
(Time Slice)

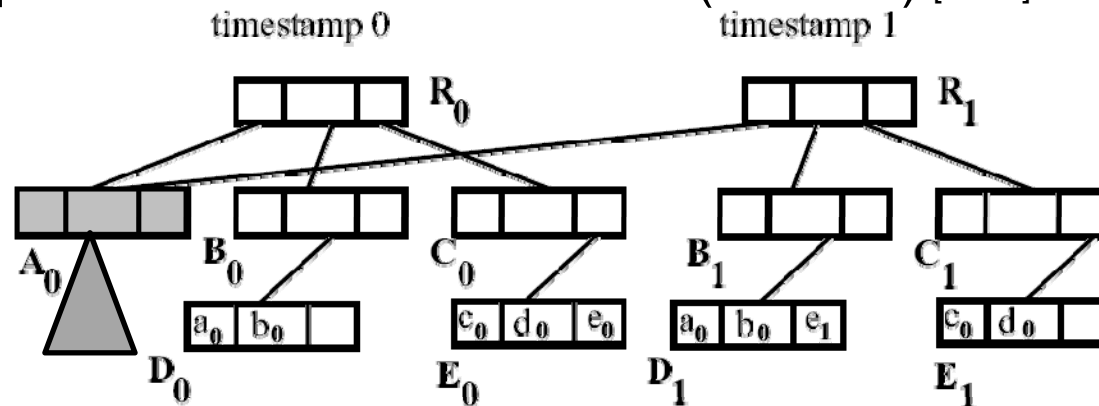


Trajektorien-basierte
Anfragen

- 3D-R-Baum skaliert gut bei Bereichsanfragen (insb. bei Time-Slice Anfragen)
- STR-Baum und TB-Baum schlagen den 3D-R-Baum generell bei geringer Anzahl von Trajektorien
- Bei Trajektorien-basierten Anfragen ist der TB-Baum deutlich besser als seine Konkurrenten.
- Bzgl. der Anfrageperformanz stellt der STR-Baum kein guten Kompromiss zwischen den 3D-R-Baum und dem TB-Baum dar.

– Multi-Version R-tree (MVR) Indexe

- **Ziel:** Gute Unterstützung von räumlichen Anfragen die sich auf einen Zeitpunkt beziehen (Time-Slice Anfragen).
- **Idee:** Separater Baum (z.B. R-tree) für jeden Zeitpunkt t .
- Beispiel: Der Historische R-Baum (HR-tree) [NS98]



- Objekt e bewegt sich von Objekten c und d in Richtung a und b
=> Dublizierung der Knoten D und E (D_1 und D_2 halten die neue Situation unter Zeitpunkt 1 fest.)
- Zusätzliche Verlinkung von Indexknoten die sich über zwei oder mehrere Zeitpunkte nicht verändern um Speicherplatz zu sparen.

– Diskussion

- 3D-R-Baum und Varianten (STR-tree, TB-tree)
 - TB funktioniert gut für Anfragen über einen größeren Zeitraum
 - Schlechtes Anfrageverhalten bei Anfragen die sich auf einen Zeitpunkt beziehen (R-Baum besser)
- Multi-Versions R-Bäume (MR, HR und MVR)
 - Funktionieren gut für Anfragen zu bestimmten Zeitpunkten (Snapshot-Anfragen)
 - Anfragebearbeitung degeneriert bei Anfragen über einen größeren Zeitraum
- Idee: Kombiniere beide Ansätze in eine Indexstruktur
 - Kombination aus 3D-R-Tree-Varianten und MVR-tree zur Unterstützung von Kurzzeit-Anfragen sowie Anfragen über einen längeren Zeitbereich
 - Implementierung dieses Ansatzes: Beispiel: MV3R-tree [TP01]

- Hybride Ansätze wie der MV3R-Baum [TP01]
 - Koppelt zwei Indexstrukturen, den 3D-R-Baum und den MVR-Baum (HR-Baum) miteinander.
 - Beide Bäume haben gemeinsame Blätter (Datenbasis)
 - Vereint die Stärken der beiden Ansätze:
 - 3D-R-Baum für Anfragen über einen gewissen Zeitintervall
 - HR-Baum für Zeitpunkt-Anfragen (Time-Slice Anfragen)
 - Prinzip:
 - Anfragen für kurze Zeitbereiche: Suche im jeweiligen MVR-Baum
 - Anfragen für längere Zeitbereiche: Suche im 3D-R-Baum falls Anfragezeitraum einen Benutzerdefinierten Grenzwert überschreitet.

