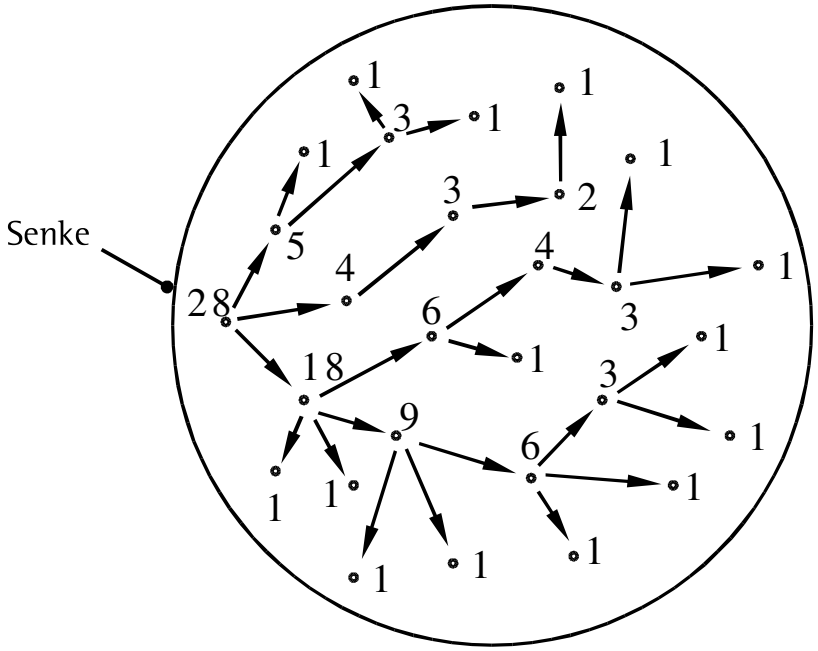
Aggregation

Tiny
Aggregation

Problem der Datenverdichtung in Richtung Senke

Jeder Knoten erfaßt einen eigenen Datenvektor und sendet diesen zusammen mit allen empfangenen in Richtung der Senke (Wurzel) des Sensornetzes weiter. Dabei akkumulieren sich die Daten immer stärker. Im schlimmsten Fall können die Knoten nahe der Senke die Daten nicht mehr aufeinmal übertragen und haben evtl. nicht genug Speicher zum Puffern. Dies würde bei großen Netzen zwangsläufig zum Datenverlust führen.

In gewissem Maß kann fehlende Bandbreite durch mehr Pufferspeicher in den Knoten ersetzt werden und eine glm. Auslastung ermöglichen. Trotzdem bleiben beide Ressourcen begrenzt.



©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Aggregation

Tiny
Aggregation

Aggregation von Daten in Sensornetzen nach TAG

„TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks“ von S. Madden, M. J. Franklin, J. M. Hellerstein

Häufig hat das Sensornetz nur einen Zugang zum Festnetz. Im folgenden wird über das Netz ein Baum aufgebaut, dessen Wurzel derjenige Knoten mit Netzzugang ist. Alle Knoten versenden Broadcast-Nachrichten. Die Adressierung geschieht mittels ID (0=Broadcast an alle).

(1) Der Root-Knoten deklariert sich selbst als Wurzel und versendet das Tripel (Baum-Level, ID, Vaterknoten)

(2) Alle Knoten innerhalb der Sendereichweite speichern die Root-ID als Vaterknoten, setzen ihr Level=VaterLevel+1 und erzeugen eine eigene ID

(3) Alle Kinderknoten „ziehen“ einen Timer mit zufälliger Ablaufzeit auf. Der Knoten der die kürzeste Ablaufzeit gewählt hat definiert sich selbst als inneren Baumknoten und verkündet per Broadcast das oben erwähnte Tripel. Mögliche Verbesserung: Die Brüder handeln denjenigen mit der besten Konnektivität aus.

(4) Alle Kinder im Empfangsbereich fahren analog zu (2) fort. Broadcast-Nachrichten werden von allen Knoten ignoriert, die bereits zuvor eine solche gehört haben.

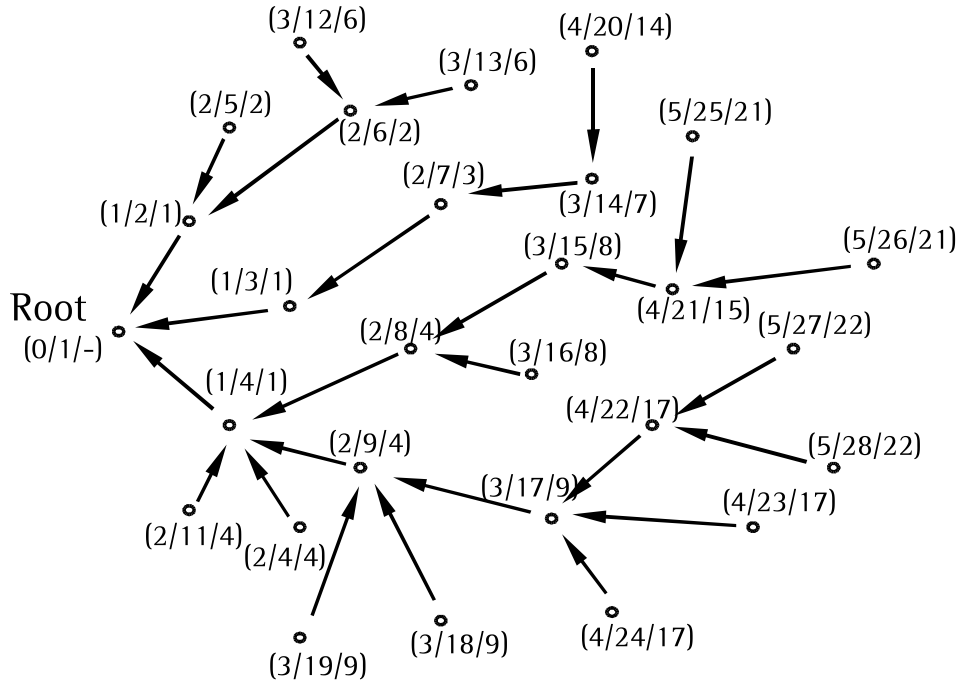
Bemerkung: Der Aufbau des Baumes ähnelt stark dem der Baum-Aufbau Phase des AMRIS Protokolls.

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Aggregation

TAG: Initialer Aufbau des Netzes

Beispiel für einen Baum über einem Sensornetz



(Level/ID/Parent)

©Thomas Haenselmann – Praktische
Informatik IV – Universität Mannheim

Aggregation

TAG: Anfragemodell

Anfragen sind syntaktisch an SQL angelehnt und werden an den Wurzelknoten gerichtet. Dieser propagiert sie dann an seine Kinder weiter etc.

```
SELECT [ALL,DISTINCT, UNIQUE]
        {const, attr, fkt(expr), arithm. expression}
    FROM sensors
    WHERE {selPreds}
    GROUP BY {attrs}
    HAVING {havingPreds}
    EPOCH DURATION i
```

SELECT wird verwendet um bestimmte Attribute (Spalten) zu aggregieren oder diese (evtl. durch arithm. Ausdrücke verändert) anzuzeigen, jedoch nur solche, die auch in GROUP BY auftauchen. Neben Attributen können auch Konstanten und Funktionen in der Select-Klausel auftauchen.

SELECT preis*100/116, preis*16/116 weist brutto-Preis und USt. getrennt aus.
SELECT TODAY, preis weist den Tagespreis zusammen mit dem Datum aus
SELECT BestNr, count(*), sum(preis) aggregiert die Preise gleicher Produkte

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Aggregation

TAG: Anfragemodell

FROM ist ein konstanter Ausdruck, mit dem das Sensornetz adressiert wird. In klassischen SQL-Anfragen können hier mehrere Tabellen angegeben werden, wodurch der outer-Join realisiert wird. Im Rahmen von TAG ist jedoch nur die „Tabelle“ sensors erlaubt.

WHERE filtert Sensorknoten bevor sie aggregiert werden. Möglich sind relationale Operatoren (light > 500 lux), BETWEEN (light BETWEEN 100 AND 500), IN (NO_NEIGHBORS IN (2,3,4)), ISNULL oder IS NULL, LIKE (z. B. SerialNo LIKE „12%“). Im Rahmen eines JOINS hat WHERE hier keine Funktion, sondern ausschließlich filternden Charakter.

GROUP BY gruppiert die Knotendaten nach bestimmten Attributen. Alle Attribute nach denen gruppiert wird, müssen auch selektiert sein. Umgekehrt müssen alle Attribute die selektiert und nicht aggregiert werden, in GROUP BY auftauchen. Die Aggregierten dagegen dürfen nicht in GROUP BY erscheinen.

```
SELECT BestNr, count(*), sum(preis)
    FROM Bestellung
    GROUP BY BestNr
```

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Aggregation

TAG: Anfragemodell

HAVING filtert ganze Gruppen, die durch GROUP BY gebildet wurden. Aggregationenauch erstrecken sich immer über alle Daten-Tupel, die sich innerhalb einer Gruppe befinden. Ergebnisse von Aggregationen können also auch dazu verwendet werden ganze Gruppen zu entfernen. Einzelne Tuppel werden dagegen in der WHERE-Klausel gefiltert.

Lediglich EPOCH DURATION ist nicht SQL-konform und beschreibt die Zeit, in der ein einzelner Knoten seine Antwort an seinen Vater weitergeleitet haben soll. Die Dauer muss mindestens ausreichen, um eine Nachricht vollständig zu versenden. In der Regel umfaßt die Dauer die Zeit zum Tätigen eigener Messungen, Weiterleiten an die Kinder und Entgegennehmen der Ergebnisse der Kinder. Ein Knoten wird also von der Zeit, die er von seinem Vater zur Verfügung gestellt bekommt die selbst benötigte Zeit abziehen (soweit vorhersehbar) und seine Kinder auffordern, mit der Restzeit auszukommen.

Im Gegensatz zu klassischen SQL-Abfragen liefert eine TAG-Abfrage einen kontinuierlichen Strom von Resultaten mit einem durch EPOCH DURATION festgelegten Takt. Aufgrund von Paketverlusten und wahrscheinlichen Störungen des Sensornetzes sind diese kontinuierlichen Daten aus Sicht der Autoren von TAG sinnvoller zu interpretieren, als eine einzelne Abfrage.

©Thomas Haenselmann – Praktische
Informatik IV – Universität Mannheim

Aggregation

TAG: Anfragemodell: Beispiel

Sensoren sind innerhalb eines Gebäudes über die Stockwerke verteilt. Jeder Raum enthält mehr als einen Sensor. Sensoren kennen ihren Raum (room) und ihr Stockwerk (floor) und die aktuelle Lautstärke (volume).

Die Anfrage soll nun alle belegte Räume auf dem 6. Stockwerk ergeben, wobei ein belegter Raum eine durchschnittliche Lautstärke > threshold hat. Die Anfrage soll alle 30 Sekunden wiederholt werden.

```
SELECT AVG(volume),room FROM sensors
    WHERE floor=6
    GROUP BY room
    HAVING AVG(volume) > thresold
    EPOCH DURATION 30s
```

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Aggregation

Tiny
Aggregation

TAG: Struktur der Aggregationsfunktionen

Aggregate setzen sich aus drei Komponenten zusammen

Merging function f(<x>,<y>): Die Merging function vereinigt zwei (vektorwertige) Teilergebnisse zu einem einzigen.

Initializer i: Der Initializer bildet den Anker der kaskadierenden Aggregation, d. h. er definiert, mit was eine rekursive Funktion am Ende der Rekursion initialisiert wird.

Evaluator e: Der Evaluator faßt die letzte Aggregation zum erwünschten Endergebnis zusammen. Evtl. handelt es sich dabei nur um die Identität, so z. B. wenn Daten über das Netz lediglich gesammelt werden.

In den Blattknoten (die selbst keine Kinder besitzen) spielt die Aggregationsfunktion meist keine Rolle. Die inneren Knoten des Baumen müssen dagegen die Ergebnisse ihrer Kinder aggregieren, inklusive ihrer eigenen Messungen.

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Aggregation

TAG: Struktur der Aggregationsfunktionen: Beispiele

Der Durchschnitt AVG über einen Sensortyp

merging function: f(<value1, count1>, <value2, count2>) = <value1+value2,
count1+count2>
initializer: <value1, 1>
evaluator: result = value/count

Das Maximum MAX über einen Sensortyp

merging function: f(value1, value2) = max{value1,value2}
initializer: <value>
evaluator: result = value

Anzahl der Sensoren

merging function: f(<value1>, <value2>) = value1 + value2
initializer: 1
evaluator: result = value

©Thomas Haenselmann – Praktische
Informatik IV – Universität Mannheim

Aggregation

TAG: Struktur der Aggregationsfunktionen: Beispiele

Histogramm über einen Sensortyp

merging function: f(<value_a1, ..., value_an>, <value_b1, ..., value_bn>) =
<value_a1+value_b1, ..., value_an+value_bn>
initializer: <0, ..., 0>[value] += 1 (Inkrementiere den Vektor an der Stelle value um 1. value ist der Messwert)
evaluator: result = <value_a1, ..., value_an>

Im Gegensatz zu den vorherigen Aggregaten geht bei den Histogrammen der Wert lediglich als Index ein – möglicherweise muss er zuvor quantisiert werden. Der Index gibt dann an, welches „Bin“ im Histogramm inkremenetiert werden muss.

Ein Histogramm ist ein Vektor, der so viele Einträge (Dimensionen) hat, wie ein Messwert Ausprägungen annehmen kann (z. B. 256 Einträge für Byte-Werte). Taucht ein Wert auf, so wird im Histogramm nur der entsprechende Eintrag (der Bin) um eins inkrementiert. Sinnvolle Anwendungen z. B. für Temperaturverteilungen, Grauwerte in Bildern etc. Messwerte mit hoher Auflösung (z. B. 32 oder 64 Bit) müssen zuvor quantisiert werden, da sonst das Histogramm schnell kaum noch speicherbar ist.

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Aggregation

TAG: Klassifikation von Aggregationsfunktionen

Ausprägungen des partiellen Status (siehe Folgeseite)

Verteilt:	Jede Untergruppe von Sensoren liefert bereits ein eigenständig interpretierbares Ergebnis
Algebraisch:	Der Wert bzw. das Aggregat einer Teilgruppe ist selbst noch nicht Ergebnis. Dieses entsteht erst am Ende durch die Evaluationsfunktion.
Holistisch:	Die Größe des Aggregates einer Teilgruppe ist proportional zu den in der Gruppe enthaltenen Daten. Eine Interpretation ist erst sinnvoll, wenn alle Daten vorliegen.
Eindeutig: (unique)	So wie holistisch, jedoch sind nur paarweise unterschiedliche Werte von Interesse.
Kontext-abhängig:	Die Größe der Daten einer Teilgruppe sind abhängig von deren Inhalten.

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Aggregation

Tiny
Aggregation

TAG: Klassifikation von Aggregationsfunktionen

Im folgenden sollen die Aggregatfunktionen bzgl. vier Dimensionen klassifiziert werden.

	MAX,MIN	COUNT,SUM	AVG	MEDIAN	COUNT DISTINCT	HISTOG.
Duplikat-empfindlich	Nein	Ja	Ja	Ja	Nein	Ja
Exemplarisch / zusammen-fassend	E	Z	Z	E	Z	Z
Monoton	Ja	Ja	Nein	Nein	Ja	?
Partieller Status	Verteilt	Verteilt	Algebraisch / Vert. (je nach Berechnung)	Holistisch	Eindeutig	Kontext-abhängig

Beim partiellen Status sind dabei 5 Ausprägungen möglich. Im Rahmen der Sensornetze kommt dieser Klassifikation besondere Bedeutung zu, da sie darüber entscheidet, welcher Aufwand bei der Übertragung entsteht oder ob die Aggregation der Daten abhängig vom Volumen und der Kapazität des Netzwerkes überhaupt möglich ist.

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Aggregation

TAG: Der Aggregationsprozess

Das Aggregationsprozess besteht aus zwei Phasen, der Distributionsphase und der Sammelphase.

Distributionsphase:

Die Anfrage wird an alle Knoten des Netzwerkes weitergeleitet bzw. zumindest an die, die sie beantworten können (es gibt auch heterogene Netze mit unterschiedlich ausgestatteten Knoten). Möglicherweise müssen jedoch auch Knoten die Anfragen aggregieren und weiterleiten, die selbst keinen geeigneten Sensor besitzen.

Verändert wird bei jeder Distribution an die Kinderknoten lediglich die EPOCH DURATION Zeit. Sie wird um den Betrag verringert, den der lokale Knoten zum messen, aggregieren und weiterleiten des Ergebnisses benötigt.

Erst nach Ablauf der Zeit, die ein Knoten braucht um eigene Messungen zu machen und die der anderen Knoten entgegen zu nehmen gibt er selbst seine Daten weiter (also nicht vorzeitig). Dies ermöglicht das Schlafschema auf der folgenden Seite.

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Aggregation

TAG: Pflege der Topologie

Nachdem der Baum initial aufgebaut wurde können Veränderungen auftreten, so z. B. durch Bewegung von Knoten, durch neue Knoten und durch ausfallende Knoten. Jeder Knoten unterhält eine kleine Liste von Nachbarknoten mit KnotenID und Qualitätsmaß.

Topologieänderungen werden bei TAG auf zwei Ebenen überwacht:

(1) Ein Knoten n überprüft die Zustellrate seines Vaters p. Dies geschieht anhand einer Sequenznummer, die von p kontinuierlich hochgezählt wird. Dadurch können Lücken erkannt werden. Auch die Zustellraten der Nachbarn aus der Nachbarliste werden aufgezeichnet. Wird die Rate zu schlecht, so wählt n einen neuen Vater p' aus der mindestens so nahe an der Wurzel ist wie p und eine bessere Zustellrate hat.

(2) Ein Knoten n hat (rel. zur Epoch-duration time) länger nichts von seinem Vater gehört. n setzt seinen eigenen Level (Entfernung zum Root-Knoten) auf Unendlich und sucht einen neuen Vater. Wenn nicht anders möglich wird auch ein Knoten mit höherem Level l gewählt. Der neuen Level l+1 wird an die Kinder propagiert. Diese müssen sich dann evtl. einen neuen Vater suchen, so als ob dieser gerade ausgefallen wäre.

Bem.: Jeder Knoten schickt pro EPOCHE nur eine Nachricht. Duplikate können so auch bei Elternwechsel nicht entstehen.

©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

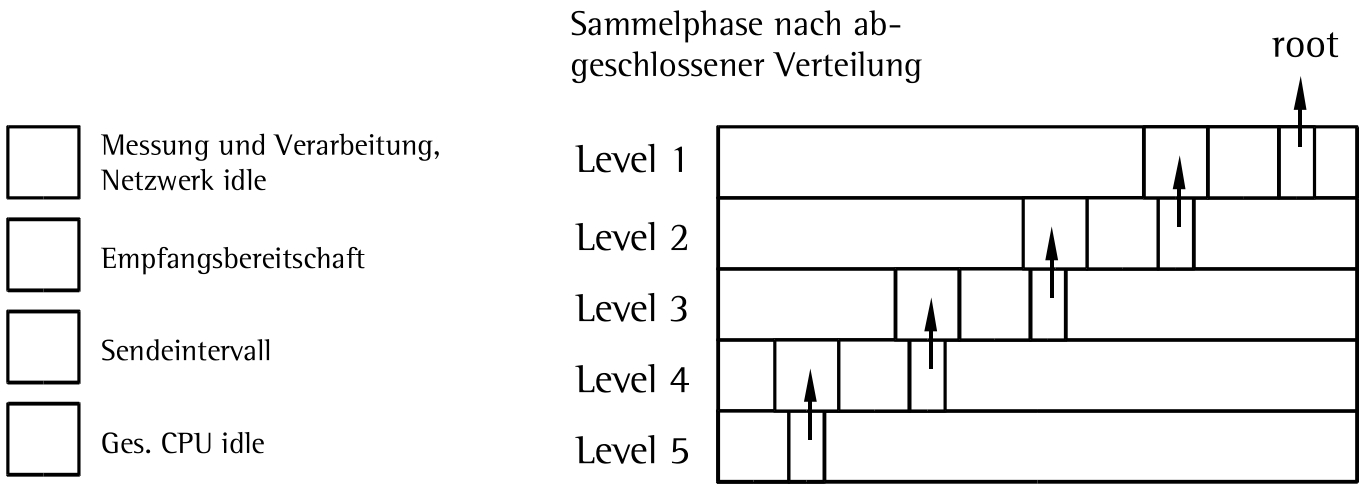
Aggregation

Tiny
Aggregation

TAG: Der Aggregationsprozess

Sammelphase:

Die meiste Zeit kann der Knoten schlafen. Während dessen muss nur die Uhr laufen, um den Knoten rechtzeitig in Empfangsbereitschaft zu versetzen. Die Empfangsbereitschaft muss die Sendephase ausreichend überlappen (im Rahmen der Ungenauigkeit der Uhren und der gegenseitigen Synchronisation). Eine Gelegenheit zur Synchronisation bietet bereits die Phase der Verteilung.

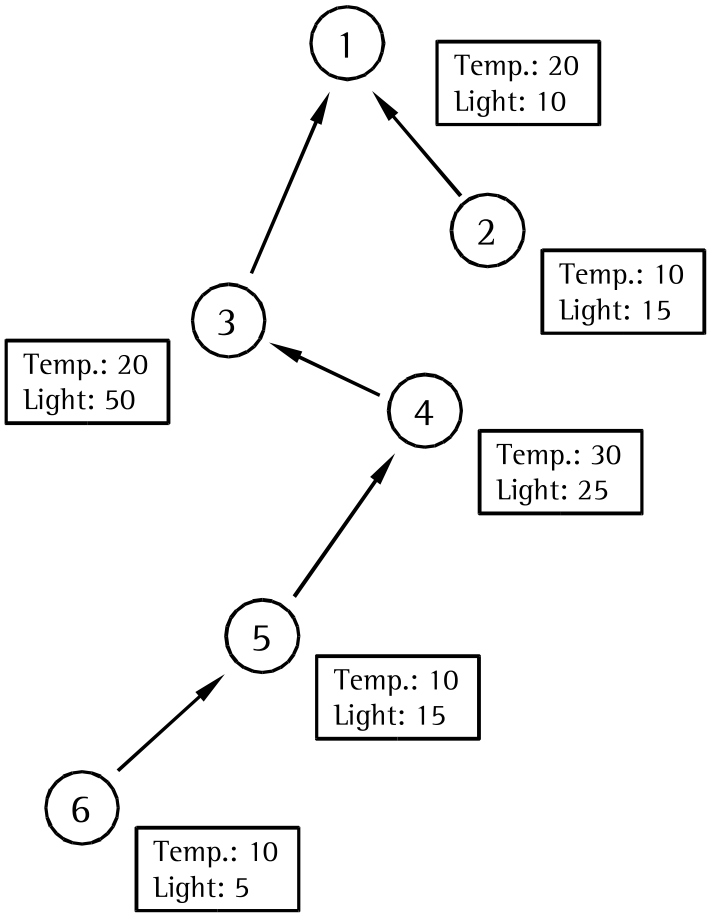


©Thomas Haenselmann – Praktische Informatik IV – Universität Mannheim

Aggregation

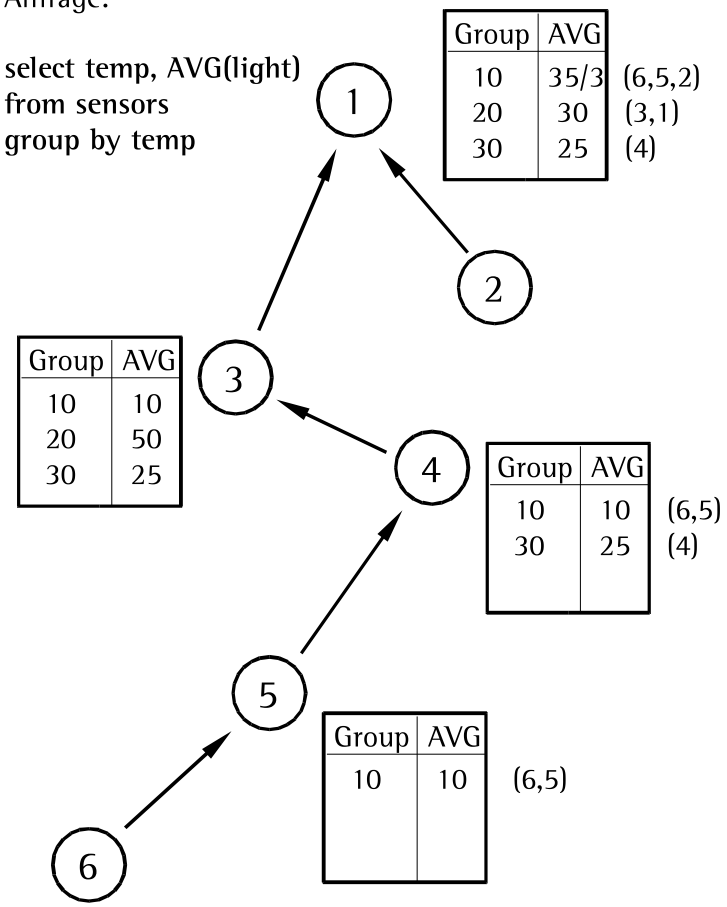
Tiny
Aggregation

TAG: Der Aggregationsprozess



Anfrage:

select temp, AVG(light)
from sensors
group by temp



©Thomas Haenselmann – Praktische
Informatik IV – Universität Mannheim