Monitoring of Machines using PostgreSQL

Roland Sonnenschein
Hesotech GmbH
automatisieren – visualisieren

http://www.hesotech.de

Wilhelm-von-Nassau-Park 11
D-65582 Diez
Germany

Location of Diez in Germany

70 km / 45 miles North-West of Frankfurt

Power Tranformer Monitoring
A Real Story

Field: 1 ... n Facilities

4 Different Monitoring-Systems

A Electrics Temps DB

B Spark-Overs DB

C Chemistry Gas in Oil DB

D Weather DB

Break-Down

Stopped 4 Weeks ago

2 Hours behind time

17 Minutes before time

In time

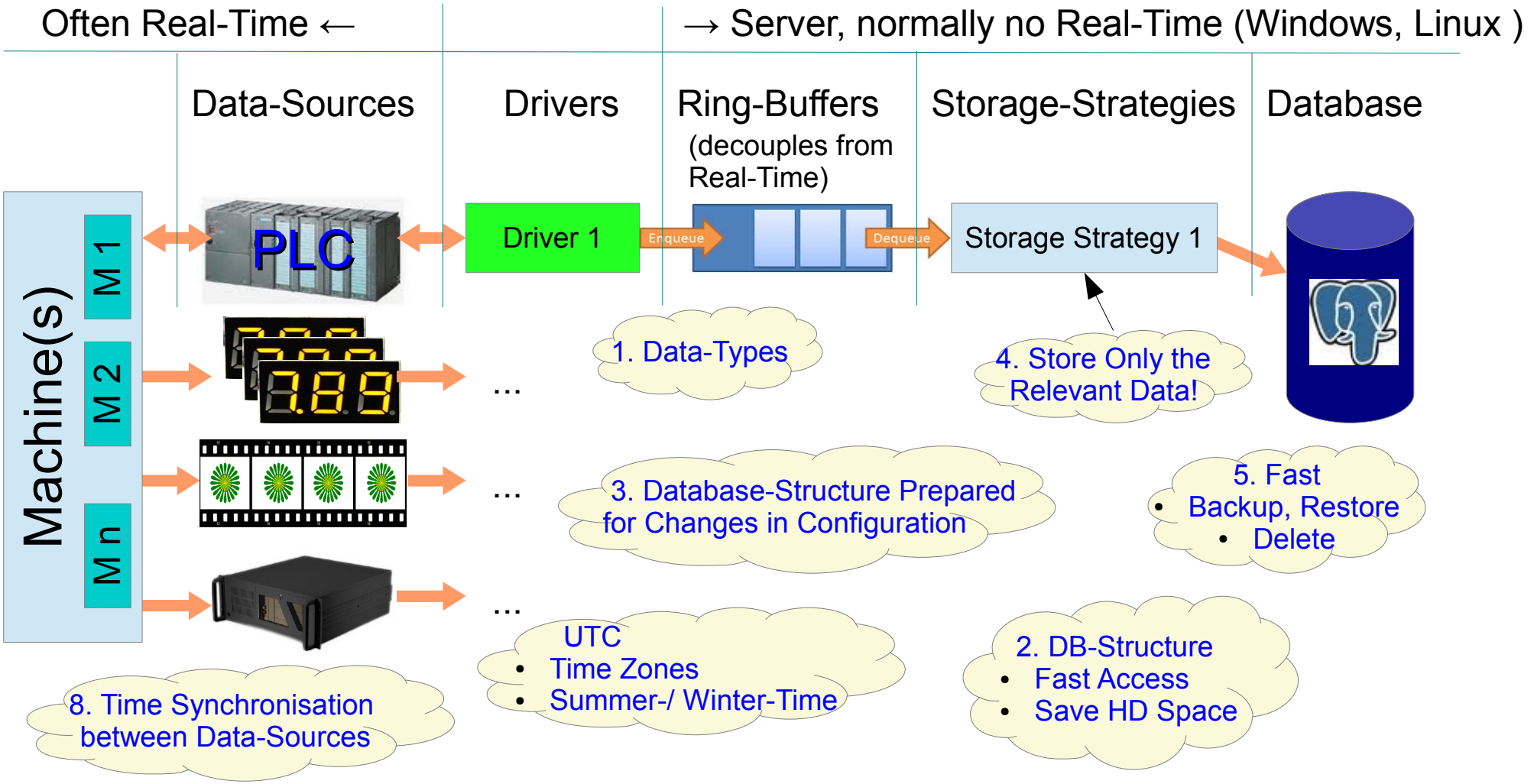
Eva-luation

How to bring data together

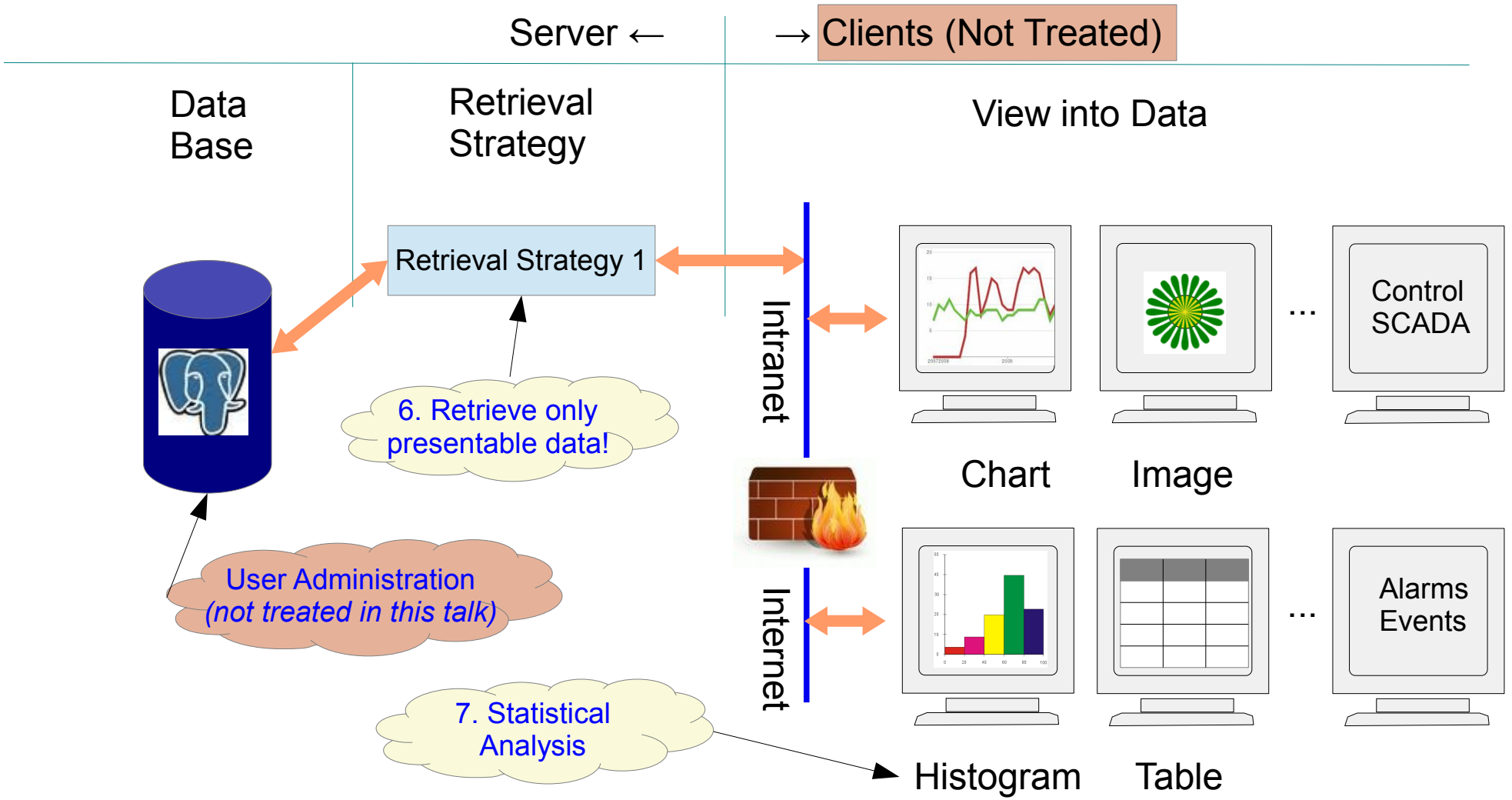
4 Programs, 4 GUIs
4 Data-Sources and Formats:
csv, proprietary, xxx-SQL, ...

Time Synchr.

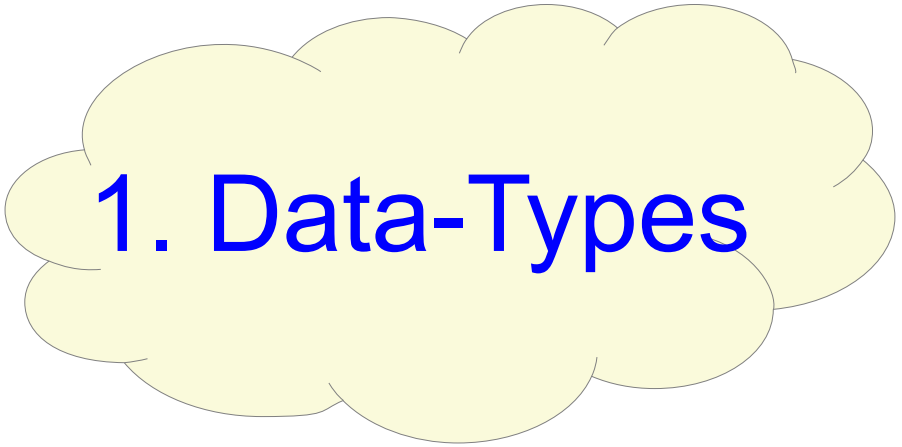
Data-Flow: Machine(s) → Database



Data-Flow: Database → User



Monitoring of Machines using PostgreSQL



- Concept of Channels
- Handling of numeric Data
- Not treated: Complex Data
  String, Image, ...

Simple Data Types (Numbers)

- Floating-Point (Resolution < 16 bit)
  Voltage, Temperature, Weight

Examples

- Integer: Value represents
  - Counter: Water-Meter
  - Enumeration: Stop, +500 U/min, - 500 U/min
  - Binary: 16 Bits in a 2 Byte Integer
- Boolean:
  - Open/Close

Uniform Storage of Numerical Data

- All numerical data are stored in the database as floating-point numbers (IEEE 754)
  - Single → 32 bit: 1 sign, 23 mantissa, 8 exponent
    No conversion losses for 16 Bit Integers !
  - Double → 64 bit: 1 sign, 52 mantissa,11 exponent
    No conversion losses for 32 Bit Integers !
- Resolution of AD Converter: <= 16 bit
- Uniform numerical storage format
  makes live **mutch easier** !
- Trick often used in SCADA systems

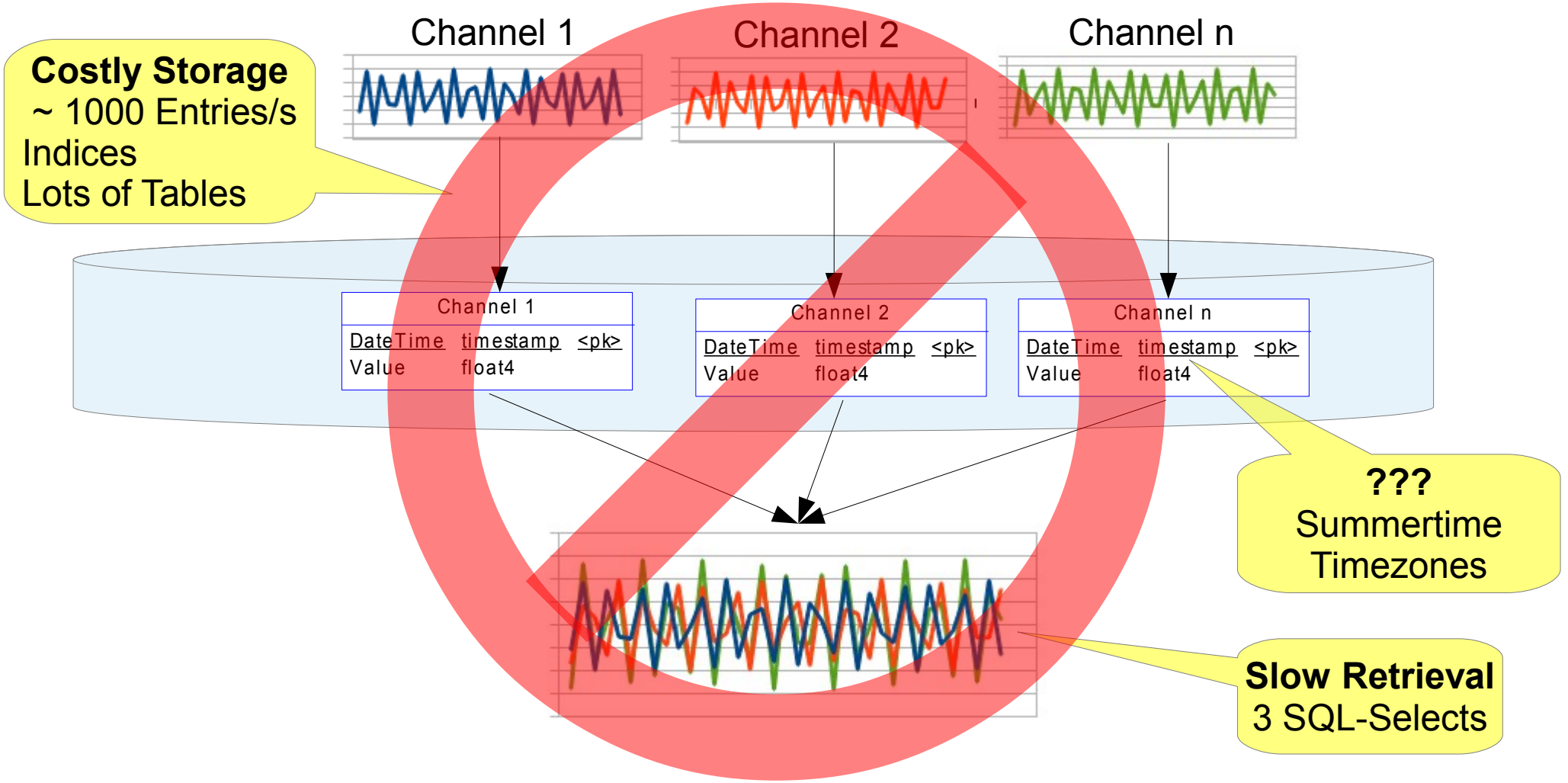
Monitoring of Machines using PostgreSQL

2. DB-Structure

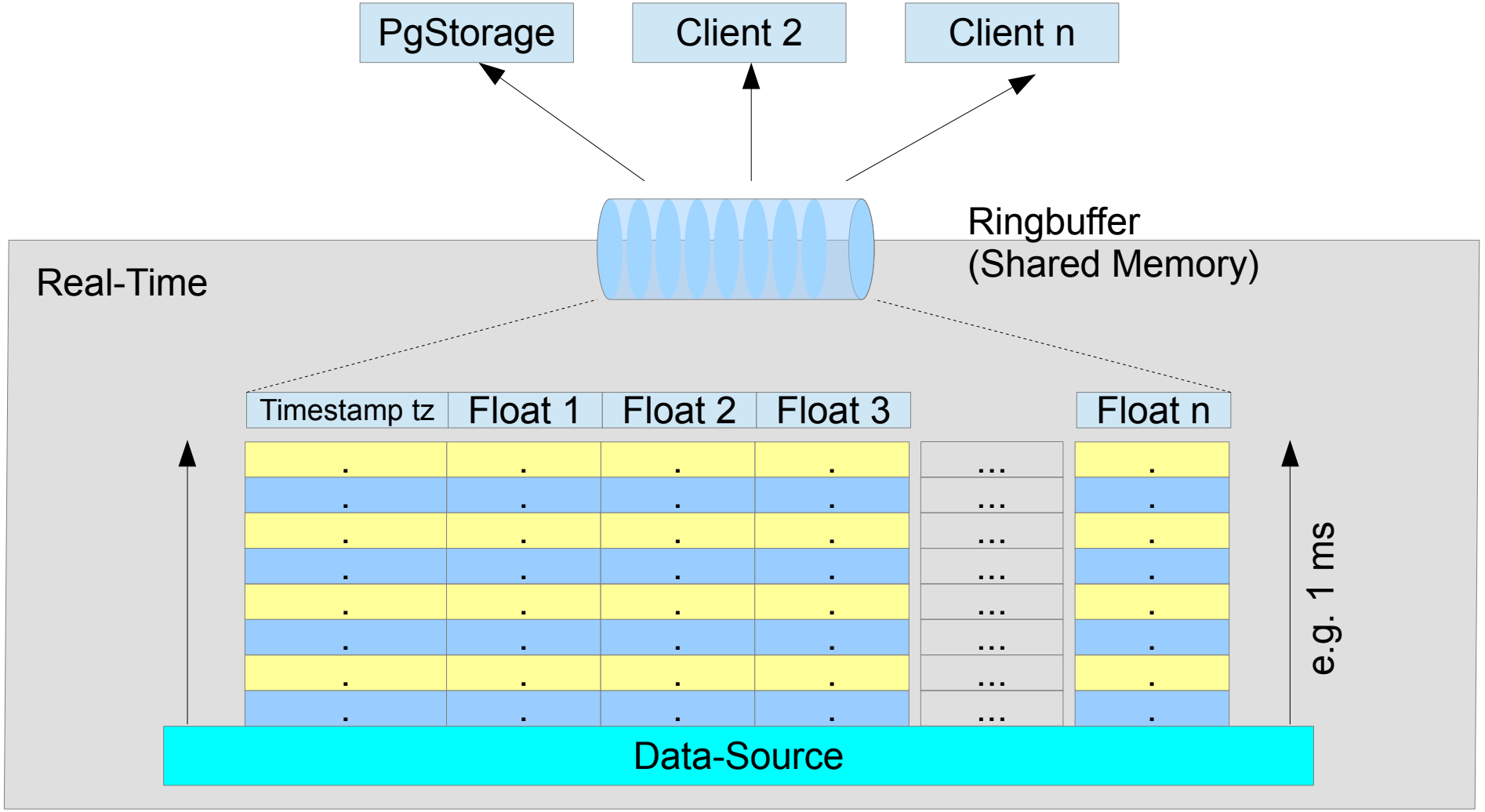
- Fast Access
- Save Harddisk Space

- Structure of Data-Tables
- Nested Storage
- Storage of Statistical Data

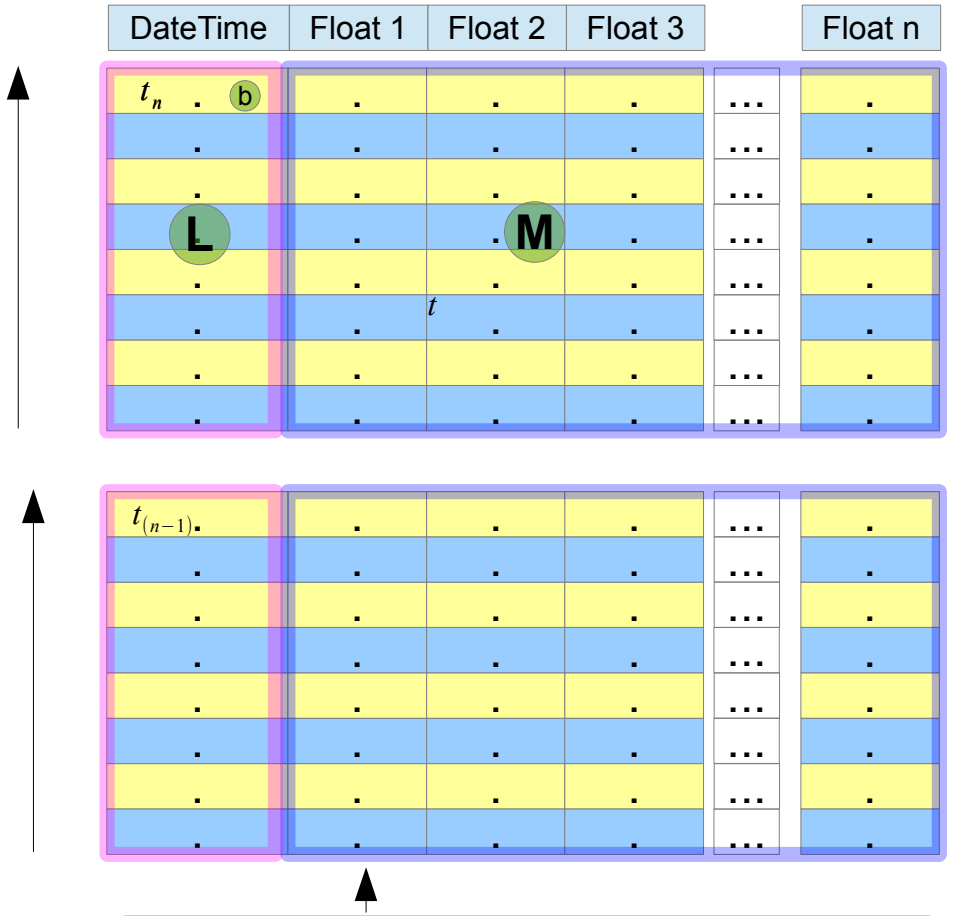
Storing Measurements (Simplest Way)



Storage: Ringbuffer



Nested Storage: Packaging



Each Package: n, min,max, avg, cur

- b • DateTime of Package
- c • $dwell = t_n - t_{(n-1)}$
- d • Count Of Measurements in Package
- e • [**min**(*Float 1*), **min**(*Float 2*), …, **min**(*Float n*)]
- f • [**max**(*Float 1*), **max**(*Float 2*), …, **max**(*Float n*)]
- g • [**avg**(*Float 1*), **avg**(*Float 2*), …, **avg**(*Float n*)]
- h • [**cur**(*Float 1*) **cur**(*Float 2*), …, **cur**(*Float n*)]
- L • List of DateTimes in Package
- M • Matrix of Measurements in Package

Valid / invalid

hischnset		
entry	bigserial	<pk>
status	INT4	
a chnsetid	INT4	
b dt_package	TIMESTAMP WITH TIME ZONE	
c dwell	INTERVAL	
d cnt_mv	INT4	
e min_mv	FLOAT4[]	
f max_mv	FLOAT4[]	
g avg_mv	FLOAT4[]	
h cur_mv	FLOAT4[]	
L dt_list	bytea	
M mv_array	bytea	

Fast statistical Evaluation

Performance of Solution

- > 3 * 10^6 Samples/s

- Example
- 300 Channels, 10 kHz
- ~ 100 Mbit/s

Monitoring of Machines using PostgreSQL

3. Database-Structure prepared for changes in Configuration

- Definition of Channel-Configurations in XML
- Handling of Configuration-Changes

Changes in Configuration

ChnSetId = 95

idx	Chn-Name	...
1	Temperatur	...
2	Voltage	...
3	Current	

insert

ChnSetId = 96

idx	Chn-Name	...
1	Temperatur	...
2	Wind	...
3	Voltage	...
4	Current	...

New Config →
Add, Remove, Change
row in pg_channelsets

Reference to Configuration

XML

hischnset

entry	bigserial	<pk>
status	INT4	
chnsetid	INT4	
dt_package	TIMESTAMP WITH TIME ZONE	
dwell	INTERVAL	
cnt_mv	INT4	
min_mv	FLOAT4[]	
max_mv	FLOAT4[]	
avg_mv	FLOAT4[]	
cur_mv	FLOAT4[]	
dt_list	bytea	
mv_array	bytea	

pg_channelsets

chnsetid	serial	<pk>
tablename	varchar(40)	
valid_from	timestamp with time zone	
valid_to	timestamp with time zone	
config	xml	

Order of channels in configuration
=
Order in float[] of table hischnset

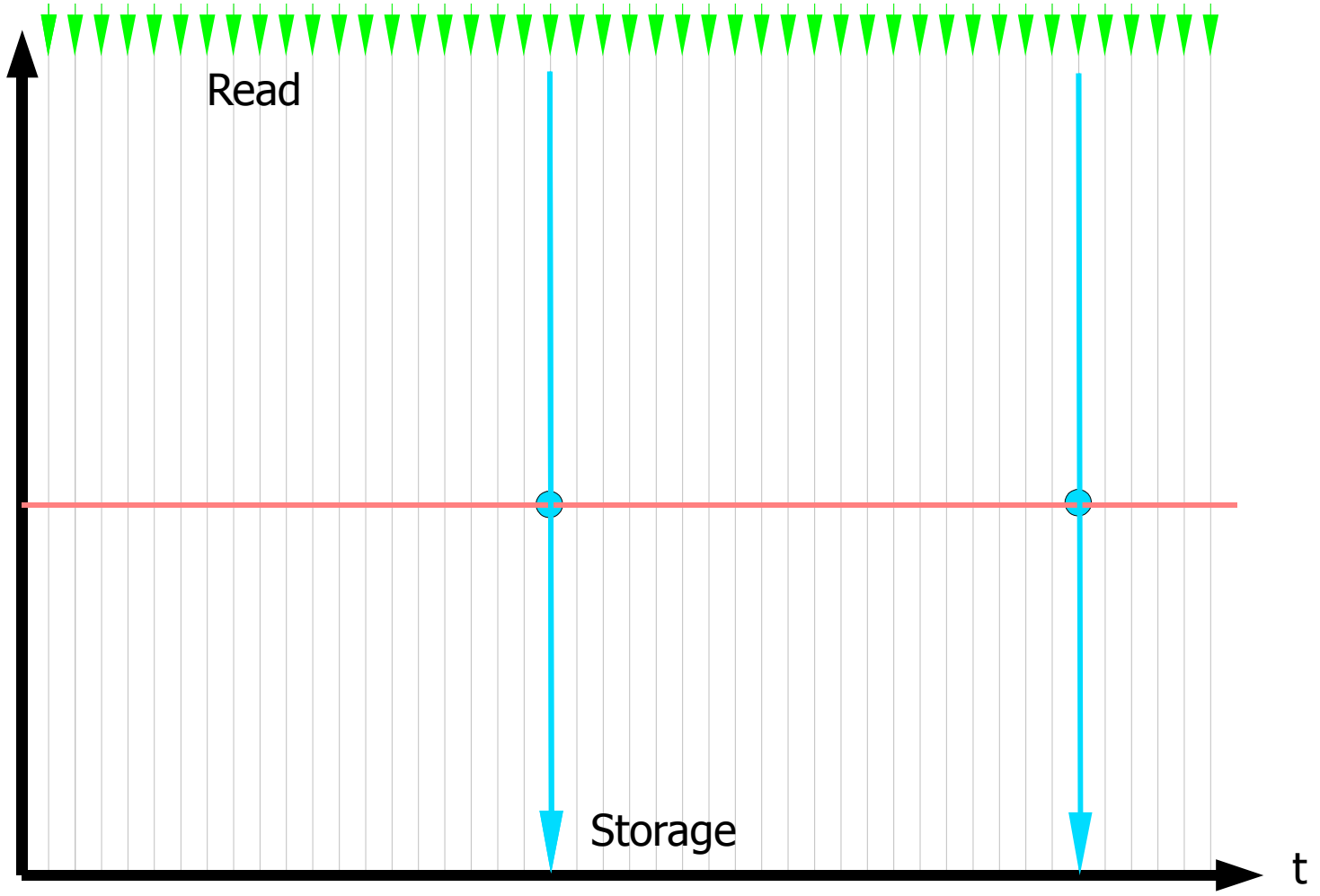
Monitoring of Machines using PostgreSQL

4. Store only the relevant data!

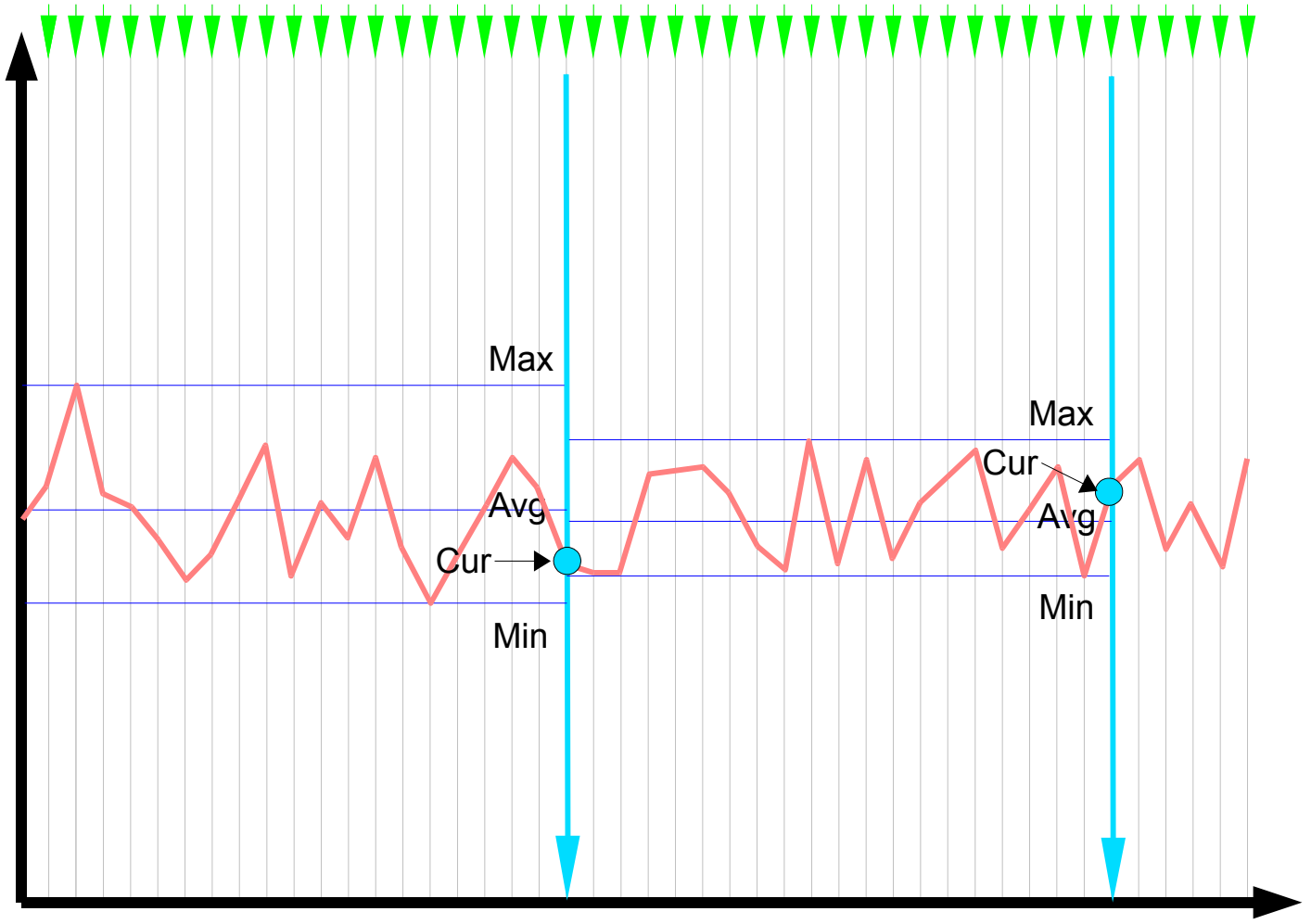
- Storage Strategies
  - ReadRate ≠ StorageRate
  - Storage Rate controlled by the Process
  - Storage on Change
  - Pretrigger

Storage Strategy (1)

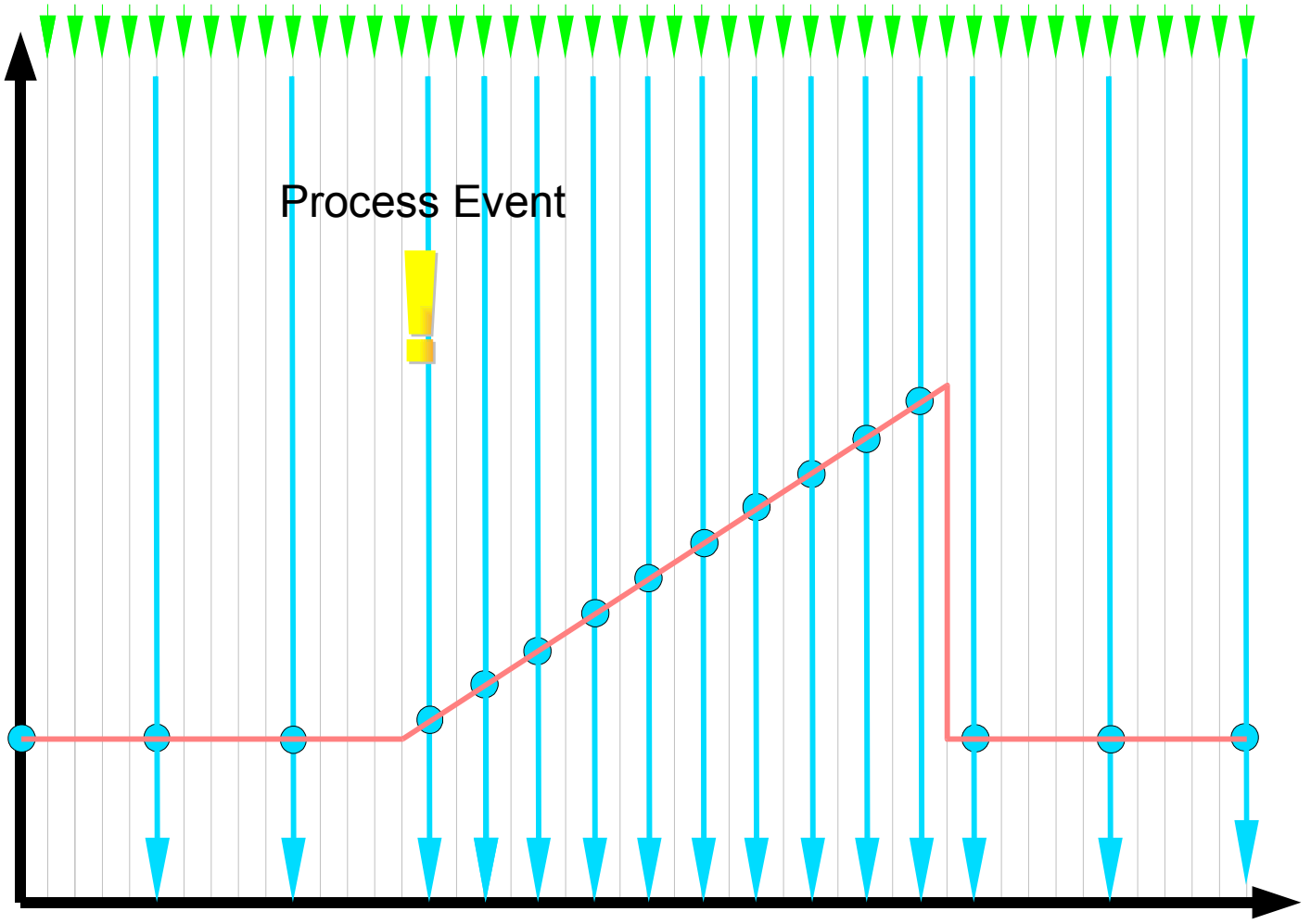
ReadRate >= StorageRate



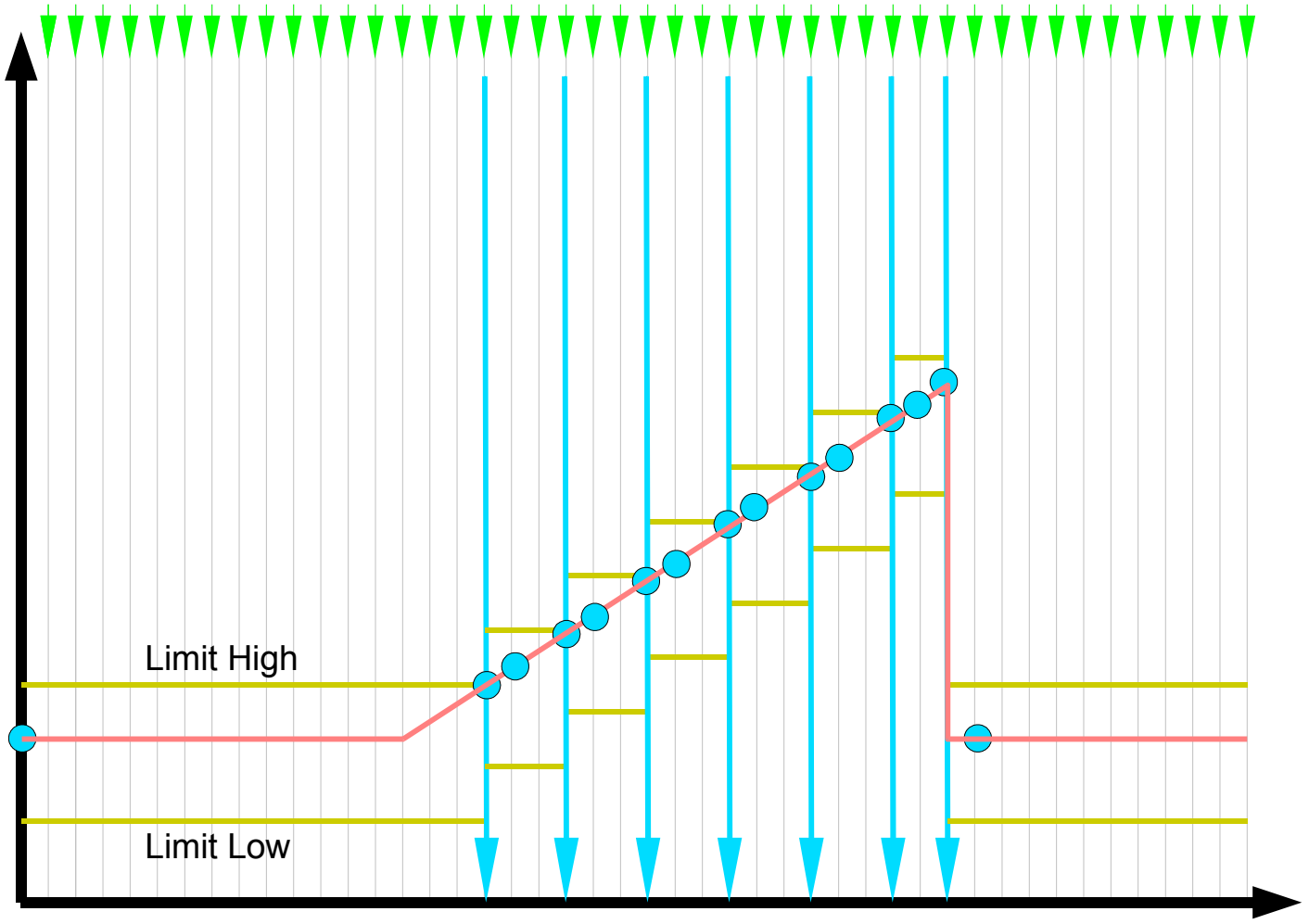
Storage Strategy (2)
Storage of Statistical Data



Storage Strategy (3)
Controlled by the Process

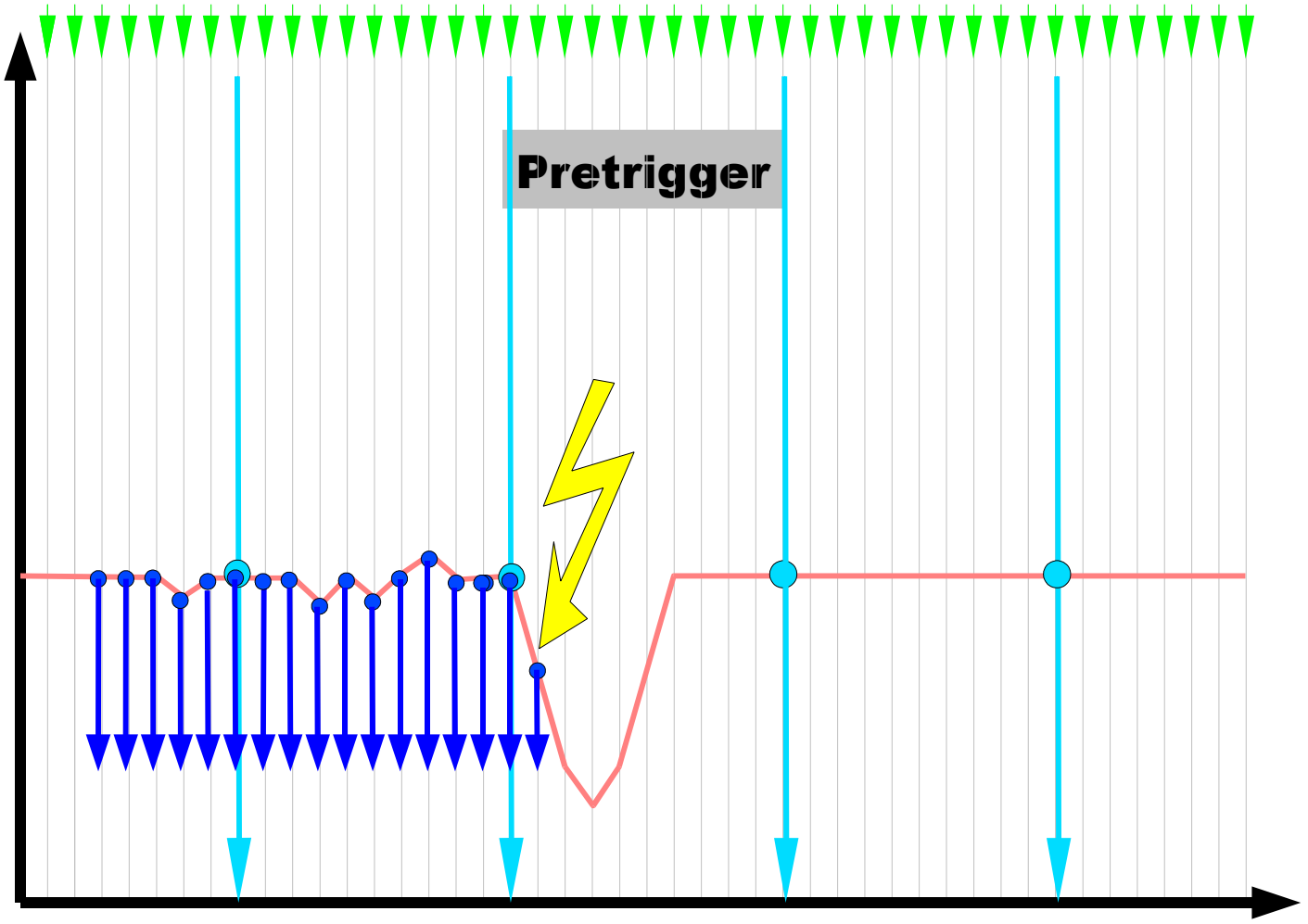


Storage Strategy (4)
Storage on Change



Storage Strategy (5)

Storage of the preliminary measurements after occurance of an Event



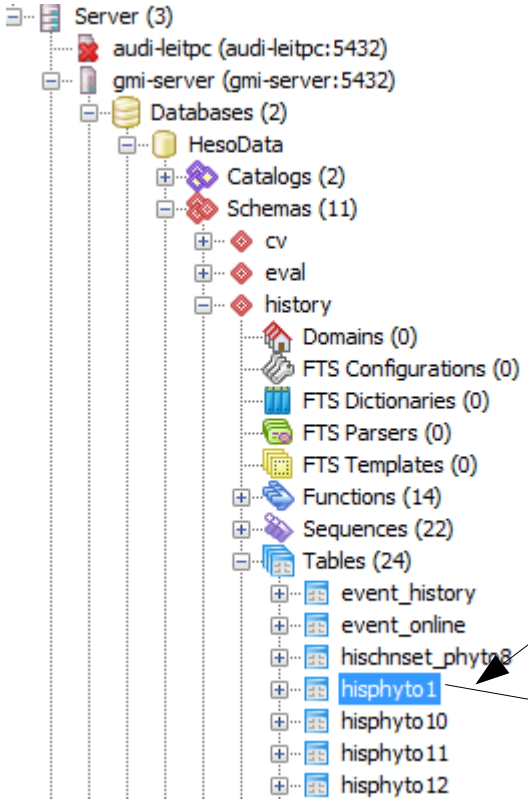
Monitoring of Machines using PostgreSQL

5. Fast
- Backup, Restore
- Delete

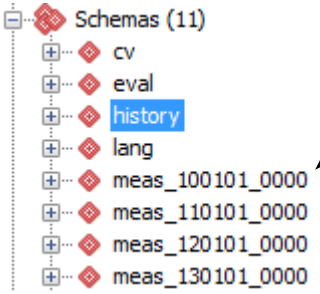
- Inheritance of Data-Tables
- Automatic generation of
  - Schemas and
  - Rules

Sectioning of the Database

One Table per Channel-Set



One Schema per Time-Period



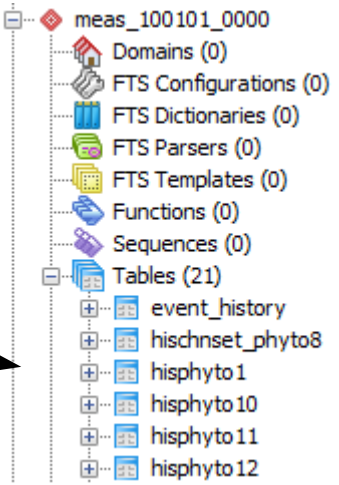
Schemas for Measurements

- Starting 2010-01-01
- Starting 2011-01-01
- Starting 2012-01-01
- Starting 2013-01-01

Empty Parent Tables

Inheritance

Tables in Schema of Time-Period are Inherited



Partitioning of Measurements

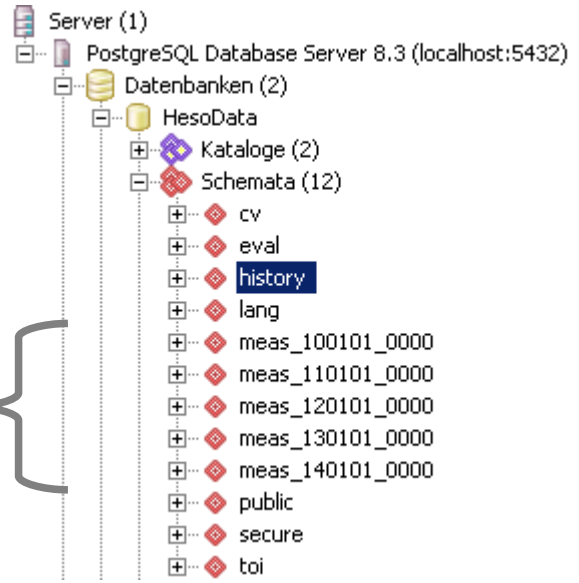
part_config		
id	int4	<pk>
part_interval	interval	
max_anz_partitions	int4	
part_prefix	text	
part_tablespace	text	

	id [PK] integer	part_interval interval	max_anz_partitions integer	part_prefix character varying	part_tablesp character var
1	1	1 year	30	meas_	pg_default

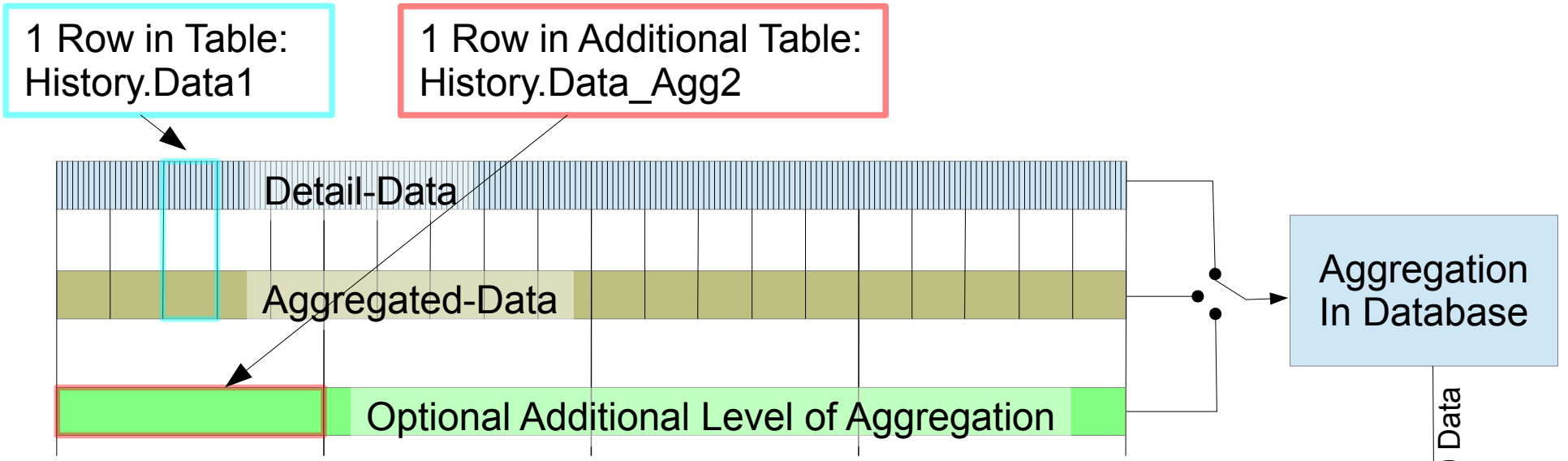
Every year one new schema created

Schema older the 30 years deleted

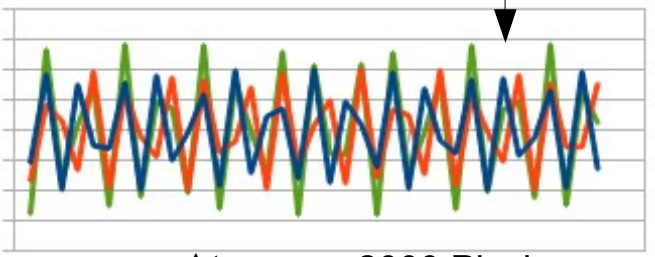
Automatic Generation Of Rules



Retrieval Strategies

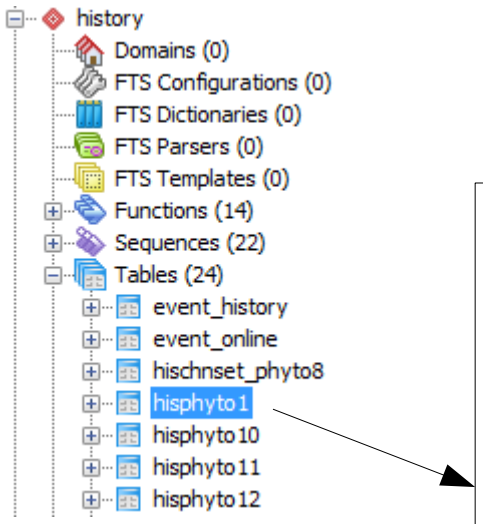


	Number of Data in Retrieval Period					
Res.	1 Sec	1 Hour	1 Day	1 Week	1 Month	1 Year
1 ms	1,0E+03	3,6E+06	8,6E+07	6,0E+08	2,6E+09	3,2E+10
1 s	1,0E+00	3,6E+03	8,6E+04	6,0E+05	2,6E+06	3,2E+07
1 min	1,7E-02	6,0E+01	1,4E+03	1,0E+04	4,3E+04	5,3E+05
3 h	9,3E-05	3,3E-01	8,0E+00	5,6E+01	2,4E+02	2,9E+03



Δt -> max 2000 Pixel

Definition of Retrieval-Strategy in Comment of Corresponding Table




```
<Parameters>
   <!--TimeSpan in C#-Notation   -->
   <!-- Timespan, to split SQL-Queries -->
   <Parameter Name="QuerySplitTime" TimeSpan="10.00:00:00" />
   <!-- Timespan between 2 entries in Detail-Data -->
   <Parameter Name="DetailReadInterval" TimeSpan="00:00:00.010" />
   <!-- Below this Timespan, Access to Detail-Data -->
   <Parameter Name="ThresholdOfDetailMode"  TimeSpan="00:10:00" />
   <!-- Above this Timespan, Acces to additional Aggregated Table -->
   <Parameter Name="ThresholdOfAggregatedTable"
                                  TimeSpan="2.00:00:00" Table="#this#_l300"/>
   <!-- Additional Threshold for aggregated tables of higher level -->
</Parameters>
```

Aggregation in SQL Query Template

```
select min(entry), min(dt_package),
#ARRAY[,,,]#          -- e.g. ARRAY[ min(min_mv[1], max(max(_mv[1])]
From {0}                                          – e.g. history.Data
where status>0                                    – is Valid
and chnsetid={1}                                  – Same Channel-Config
and dt_package >='{2}'::timestamptz               – From
and dt_package < '{3}'::timestamptz               – To
group by floor(EXTRACT(EPOCH FROM dt_package)/#GroupRangeSec#)
order by 2
```

hischnset		
entry	bigserial	<pk>
status	INT4	
chnsetid	INT4	
dt_package	TIMESTAMP WITH TIME ZONE	
dwell	INTERVAL	
cnt_mv	INT4	
min_mv	FLOAT4[]	
max_mv	FLOAT4[]	
avg_mv	FLOAT4[]	
dt_list	bytea	
mv_array	bytea	

Retrieve requested Data
as an Array.

Meaning of Array-Indizes
in ChnSetId

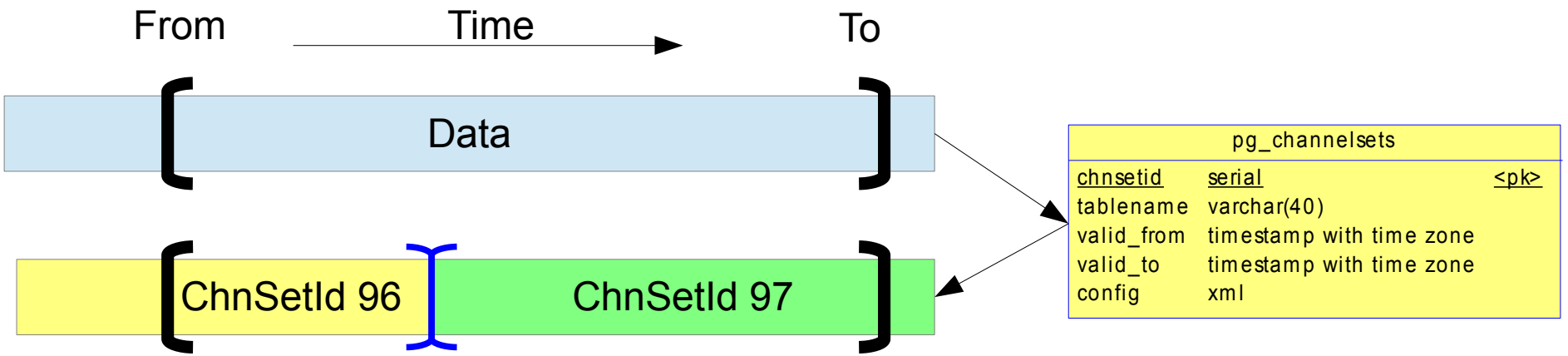
Aggregation: GroupRangeSec

```
SELECT entry,
       dt_package, EXTRACT(EPOCH FROM dt_package), floor(EXTRACT(EPOCH FROM dt_package)/600)
  FROM history.hisphyto1
  where dt_package between '2011-02-01 00:00:00' and '2011-02-02 01:00:00'
  order by 4
```

	entry bigint	dt_package timestamp with time zone	date_part double precision	floor double precision
1	235338	2011-02-01 00:00:00+01	1296514800	2160858
2	235339	2011-02-01 00:02:00+01	1296514920	2160858
3	235340	2011-02-01 00:04:00+01	1296515040	2160858
4	235341	2011-02-01 00:06:00+01	1296515160	2160858
5	235342	2011-02-01 00:08:00+01	1296515280	2160858
6	235343	2011-02-01 00:10:00+01	1296515400	2160859
7	235344	2011-02-01 00:12:00+01	1296515520	2160859
8	235345	2011-02-01 00:14:00+01	1296515640	2160859
9	235346	2011-02-01 00:16:00+01	1296515760	2160859
10	235347	2011-02-01 00:18:00+01	1296515880	2160859
11	235348	2011-02-01 00:20:00+01	1296516000	2160860
12	235349	2011-02-01 00:22:00+01	1296516120	2160860
13	235350	2011-02-01 00:24:00+01	1296516240	2160860
14	235351	2011-02-01 00:26:00+01	1296516360	2160860
15	235352	2011-02-01 00:28:00+01	1296516480	2160860
16	235353	2011-02-01 00:30:00+01	1296516600	2160861
17	235354	2011-02-01 00:32:00+01	1296516720	2160861

GroupRangeSec = 10 min

Retrieving Data: Handling Changes in Config



e.g. Temperature

```
Select Datetime, float[18] From his_table Where chnsetid=96
                                          And Datetime between .....
UNION ALL
Select Datetime, float[19] From his_table Where chnsetid=97
                                          And Datetime between .....
Order by 1
```

Monitoring of Machines using PostgreSQL

7. Statistical Analysis

- Staying Time
- Integration
- Cost Efficiency

Staying Time

- How long in 2011 did the Windmill generate more 0.7 MW ?
  - avg_mv[7] : Generated Power

```
Select sum(dwell)
    From his_table
    Where avg_mv[7]>=0.7
    And datetime between ...
```


Integration

- How long in 2011 did the Windmill generate more 0.7 MW and how much energy was generated in this status?
  - avg_mv[7] : Generated Power

$$E = \int_{t1}^{t2} P(t)\,dt$$

```
Select sum(dwell), sum(float[7]*dwell)
From his_table
Where avg_mv[7]>=0.7
And datetime between ...
```


Cost Effectiveness

- How long in 2011 was power > 0.7 M, how much energy was generated and what was the mechanical wear?
  - avg_mv[7] : Generated Power, avg_mv[9]: Velocity of Wind

```
Select sum(dwell), sum(float[7]*dwell),
sum(dwell, func_wear( avg_mv[9] )
From his_table
Where float[7]>=0.7
And datetime between ...
```

~ Earned Money

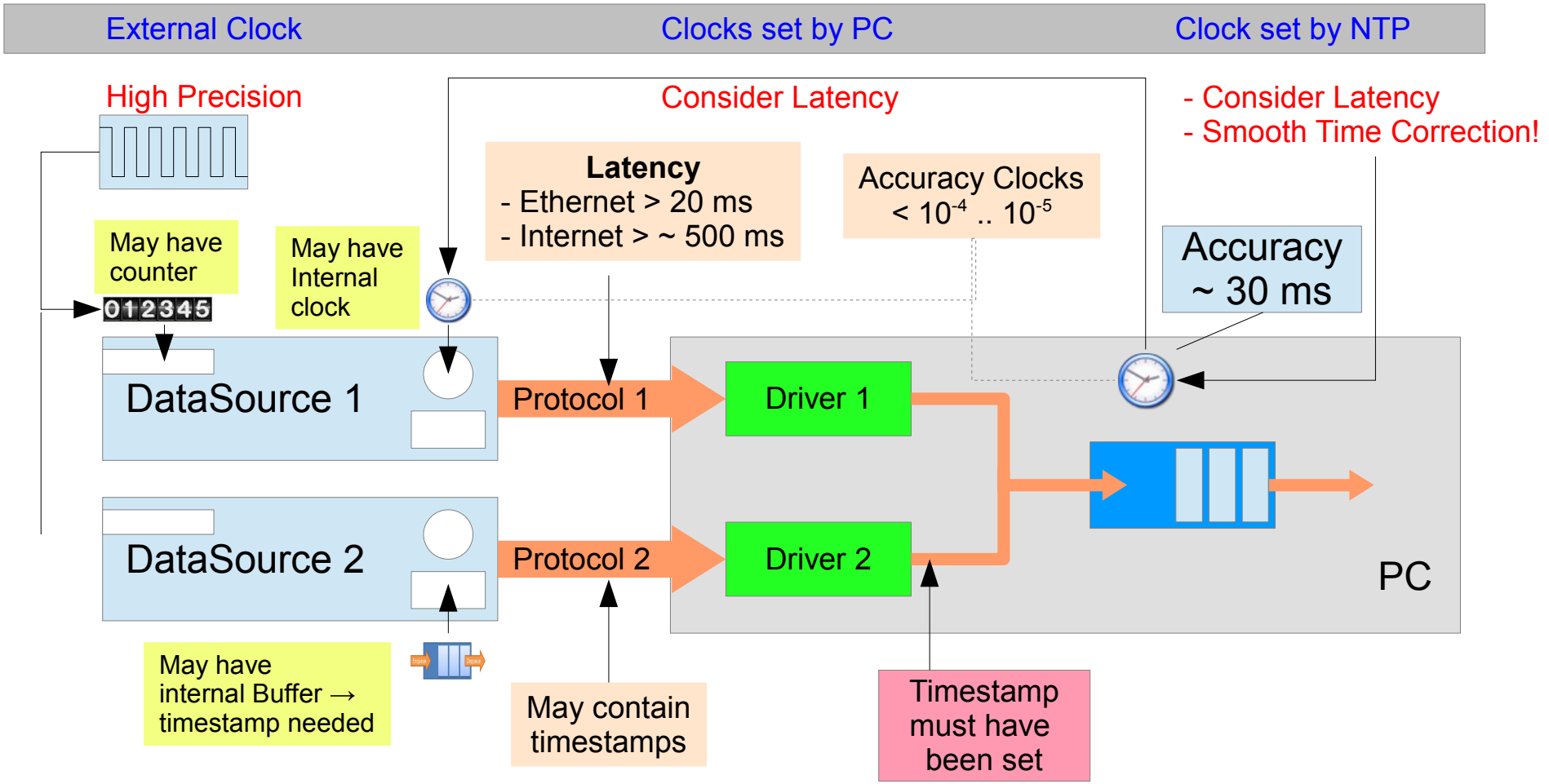
~ Cost

Livetime Consumption

Monitoring of Machines using PostgreSQL

8. Time Synchronisation between Data-Sources

Time-Synchronisation between Data-Sources: Some considerations



Time-Synchronisation Rules of Thumb

- Read-Rate
  - >~ 1 s: No Problem
  - <~ 100 ms: Some Effort
  - <~ 1 ms: (SW-) Controller for Synchronisation needed
  - <~ 0.1 ms: High Effort