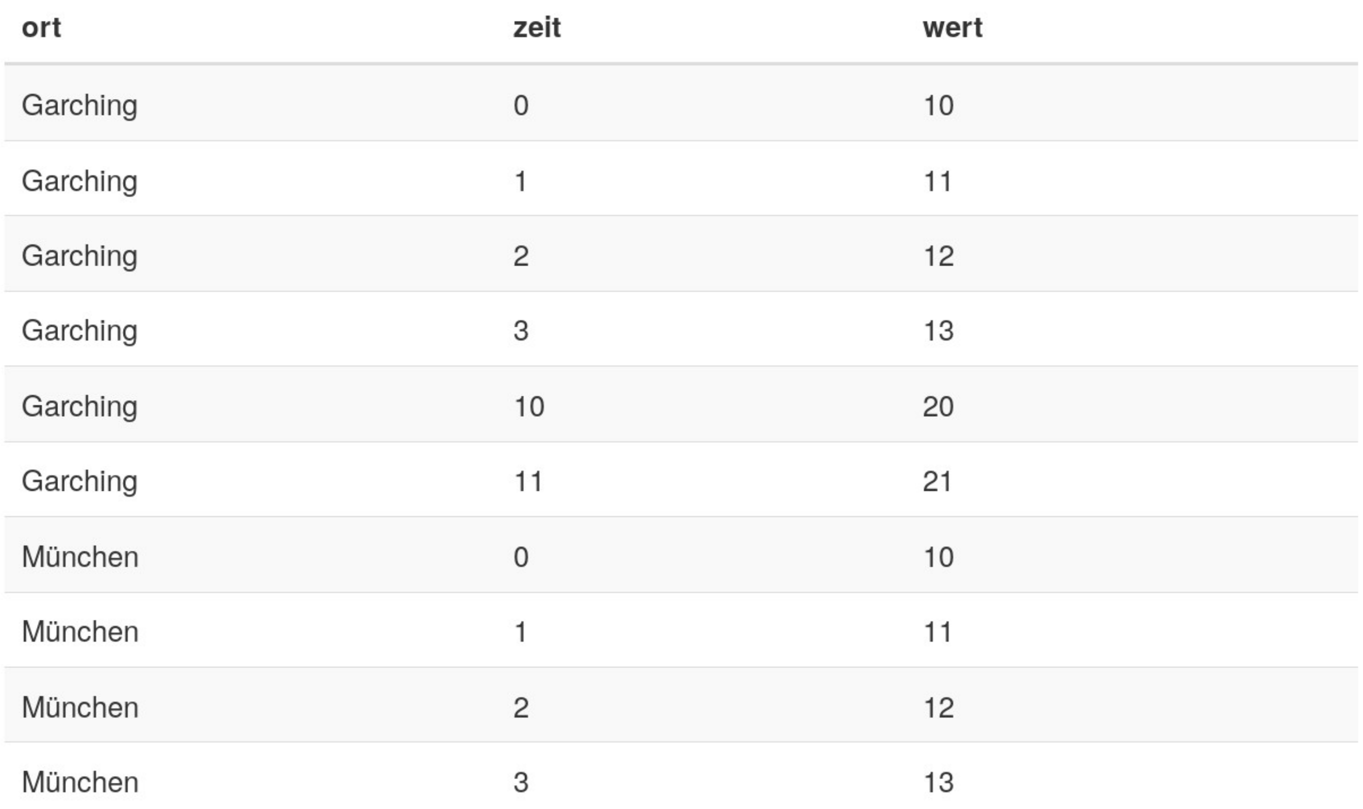
ERDB – Fensterfunktionen (Window-Functions)

Maximilian E. Schüle
i3erdb@in.tum.de
Garching, 23. Mai 2019

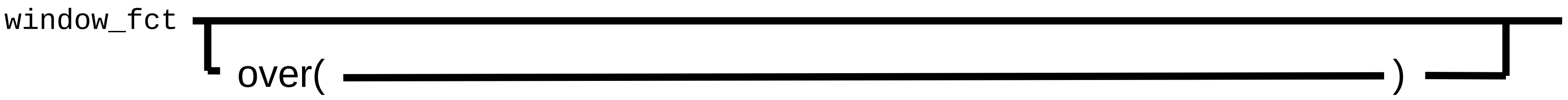
Window-Functions für Zeitreihen

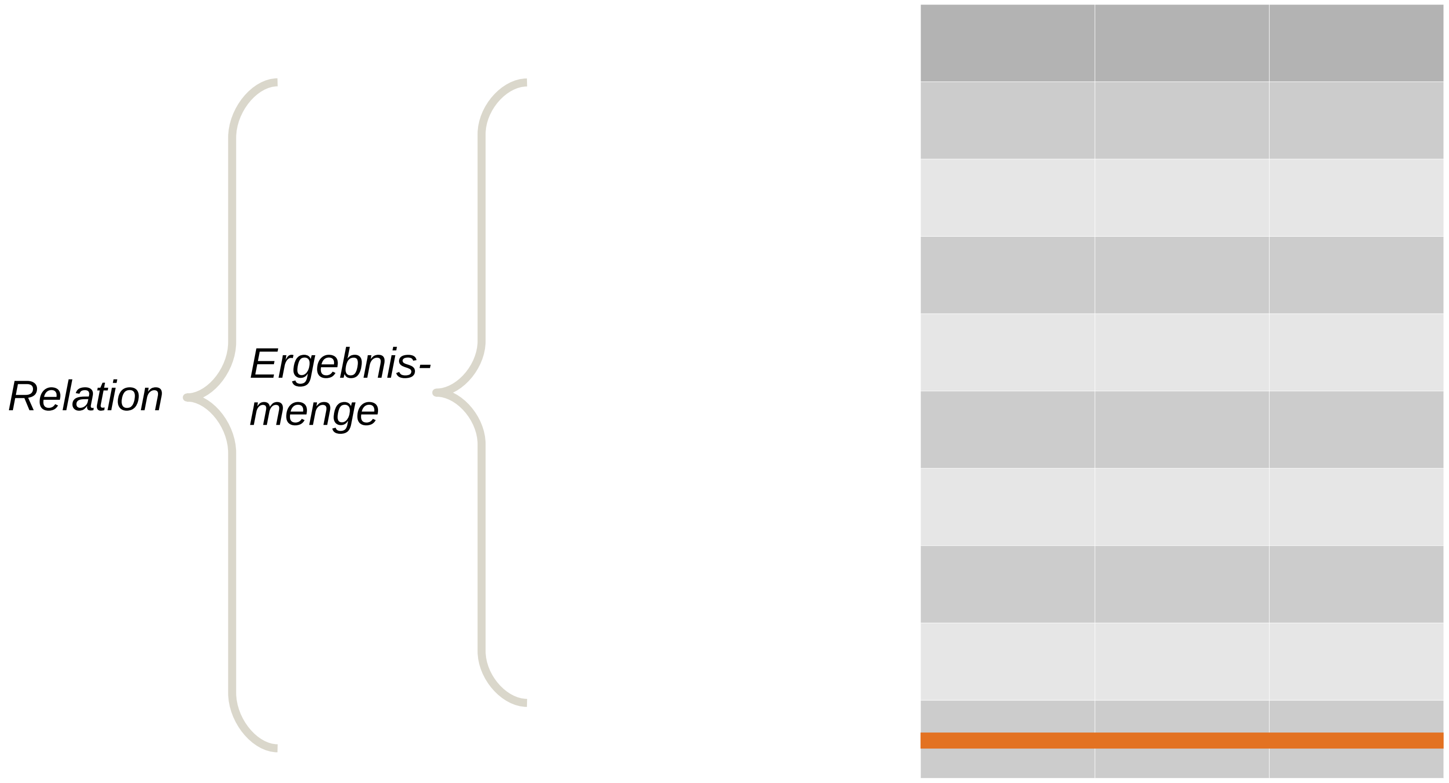
```
select *
from messungen order by Ort, Zeit
```



ort	zeit	wert
Garching	0	10
Garching	1	11
Garching	2	12
Garching	3	13
Garching	10	20
Garching	11	21
München	0	10
München	1	11
München	2	12
München	3	13

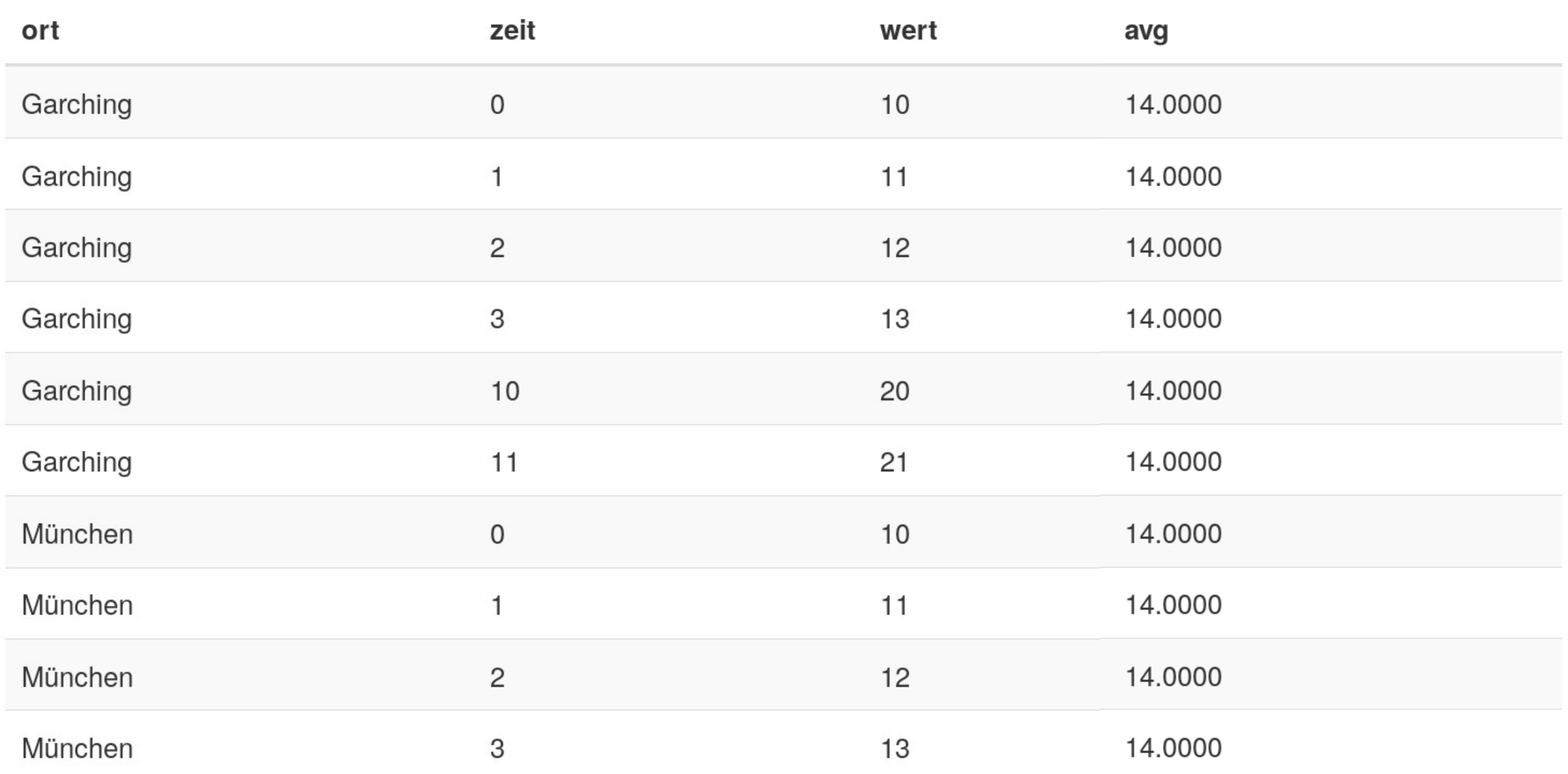
Window-Functions





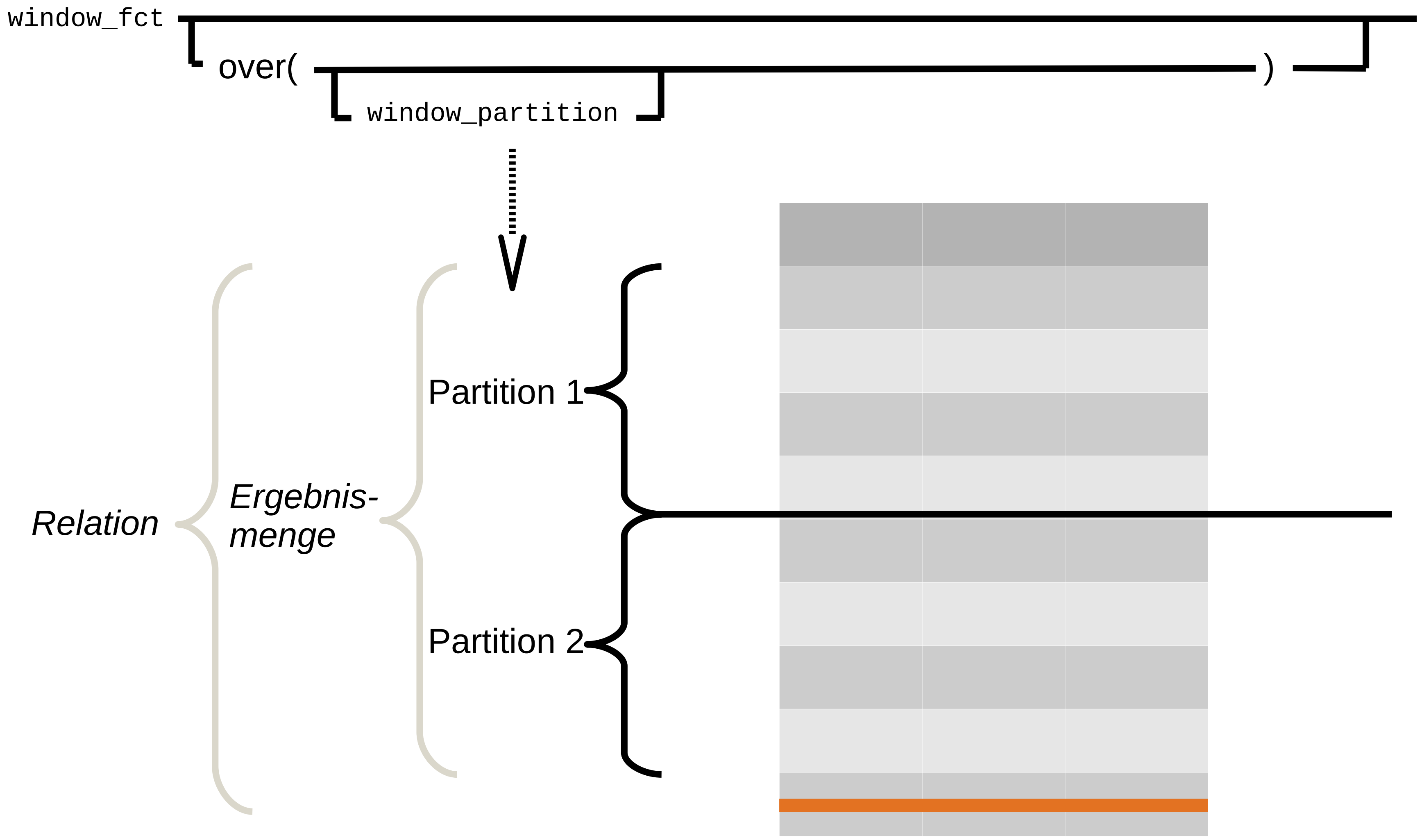
Window-Functions

```
select *, avg(Wert) over ()
from messungen order by Ort, Zeit
```



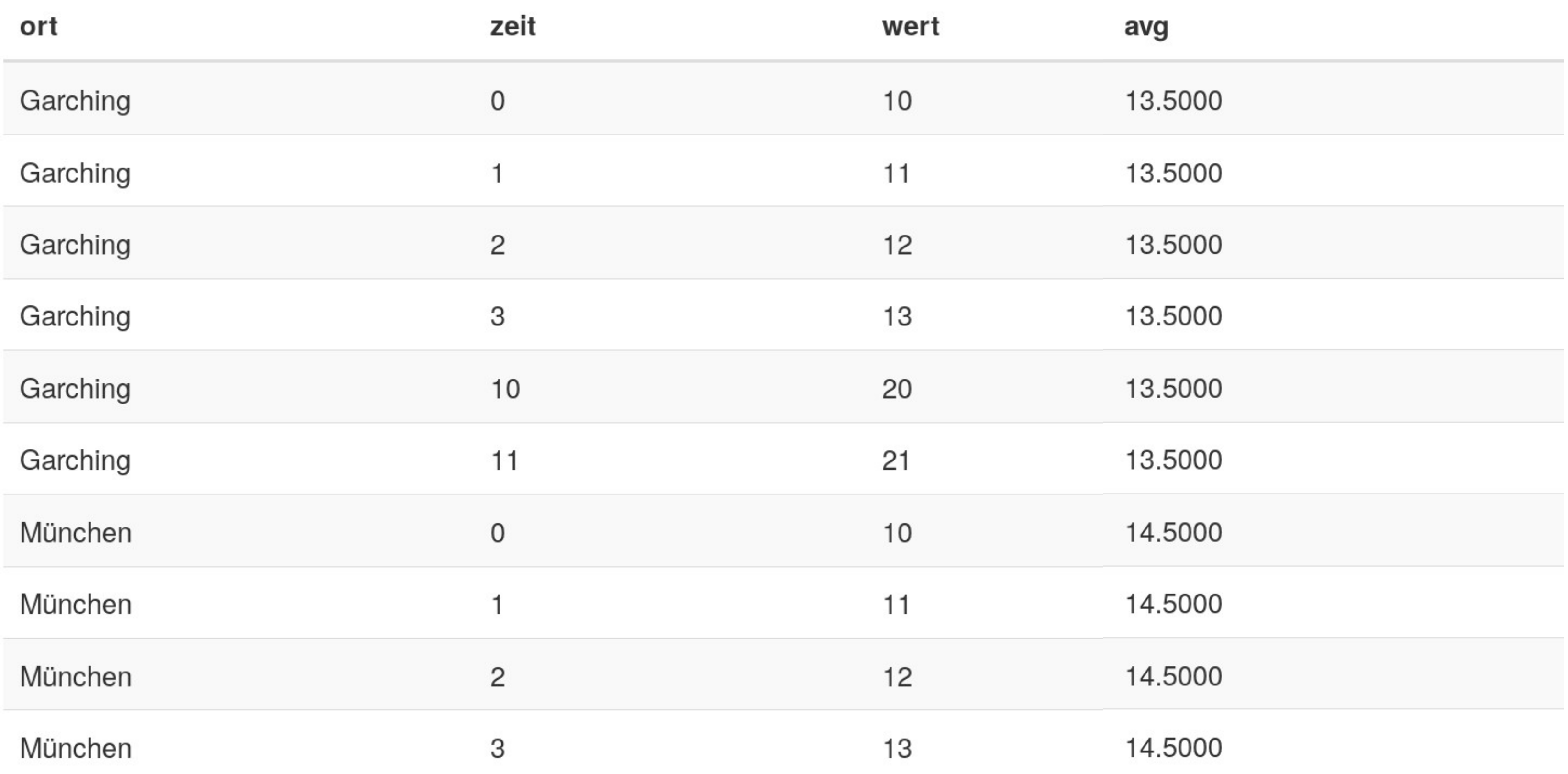
ort	zeit	wert	avg
Garching	0	10	14.0000
Garching	1	11	14.0000
Garching	2	12	14.0000
Garching	3	13	14.0000
Garching	10	20	14.0000
Garching	11	21	14.0000
München	0	10	14.0000
München	1	11	14.0000
München	2	12	14.0000
München	3	13	14.0000

Window-Functions: Partitions



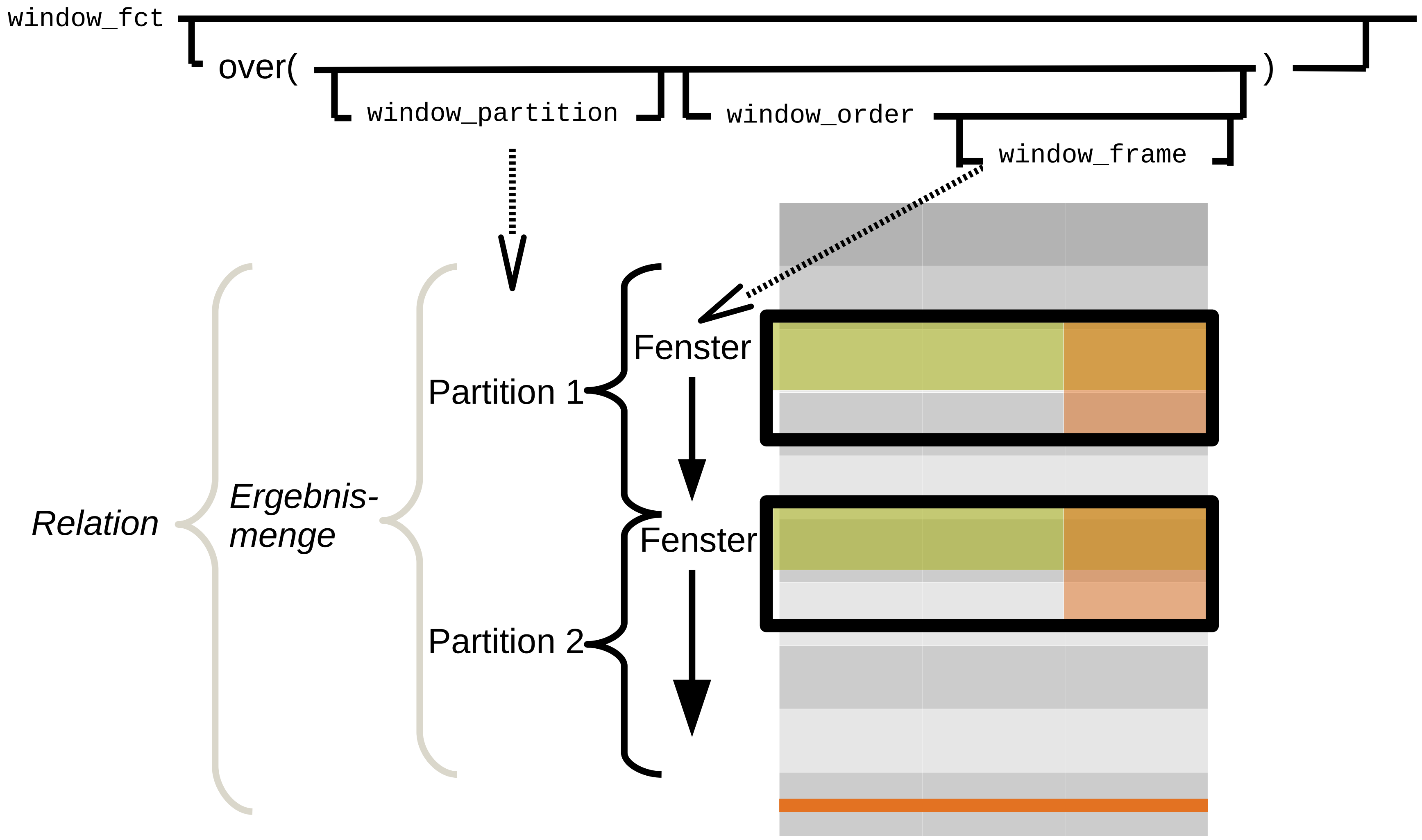
Window-Functions: Partitions

```sql
select *, avg(Wert) over (partition by ort)
from messungen order by Ort, Zeit
```



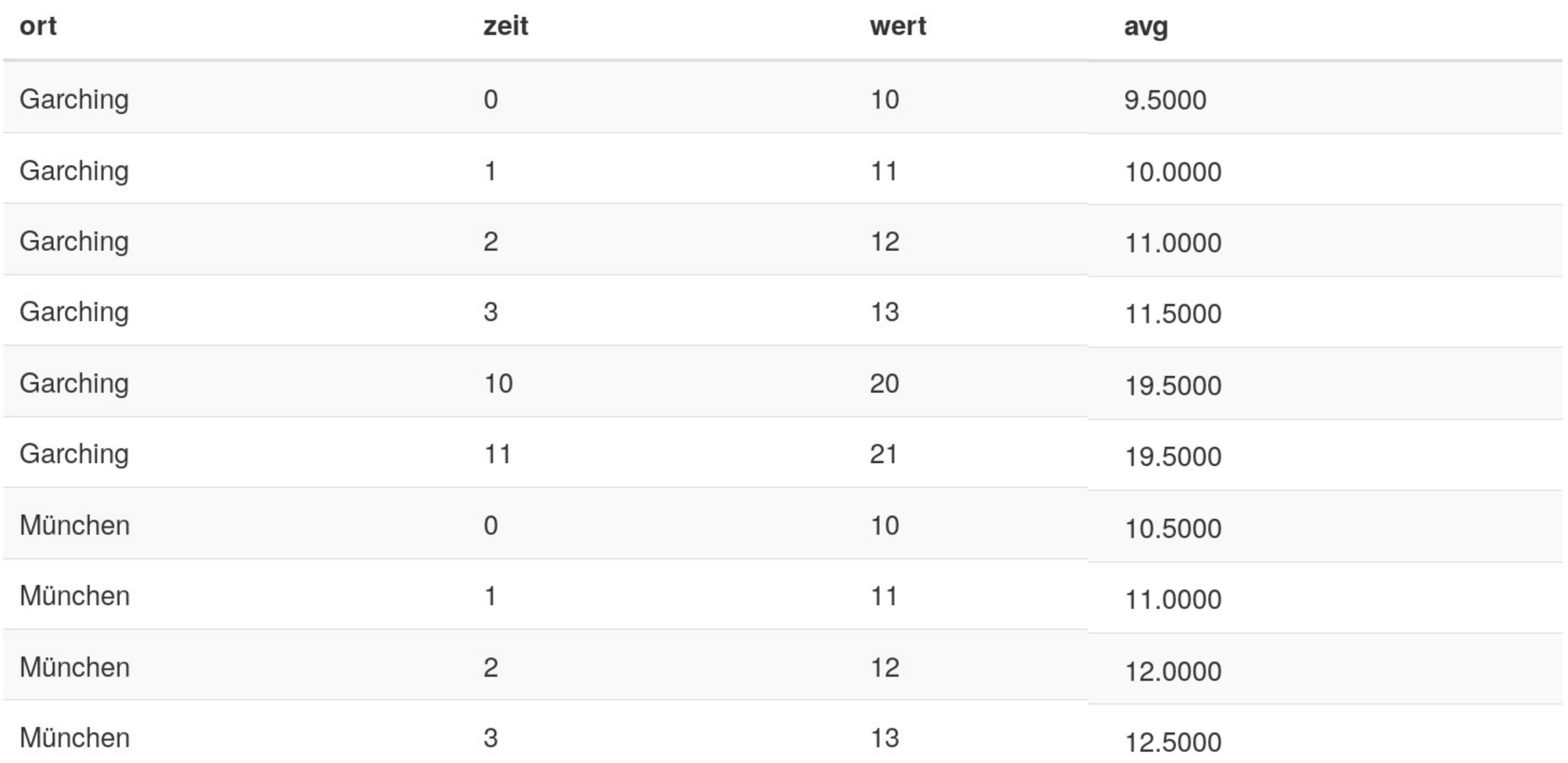
ort	zeit	wert	avg
Garching	0	10	13.5000
Garching	1	11	13.5000
Garching	2	12	13.5000
Garching	3	13	13.5000
Garching	10	20	13.5000
Garching	11	21	13.5000
München	0	10	14.5000
München	1	11	14.5000
München	2	12	14.5000
München	3	13	14.5000

Window-Functions: Order und Frames



Window-Functions: Order und Frames

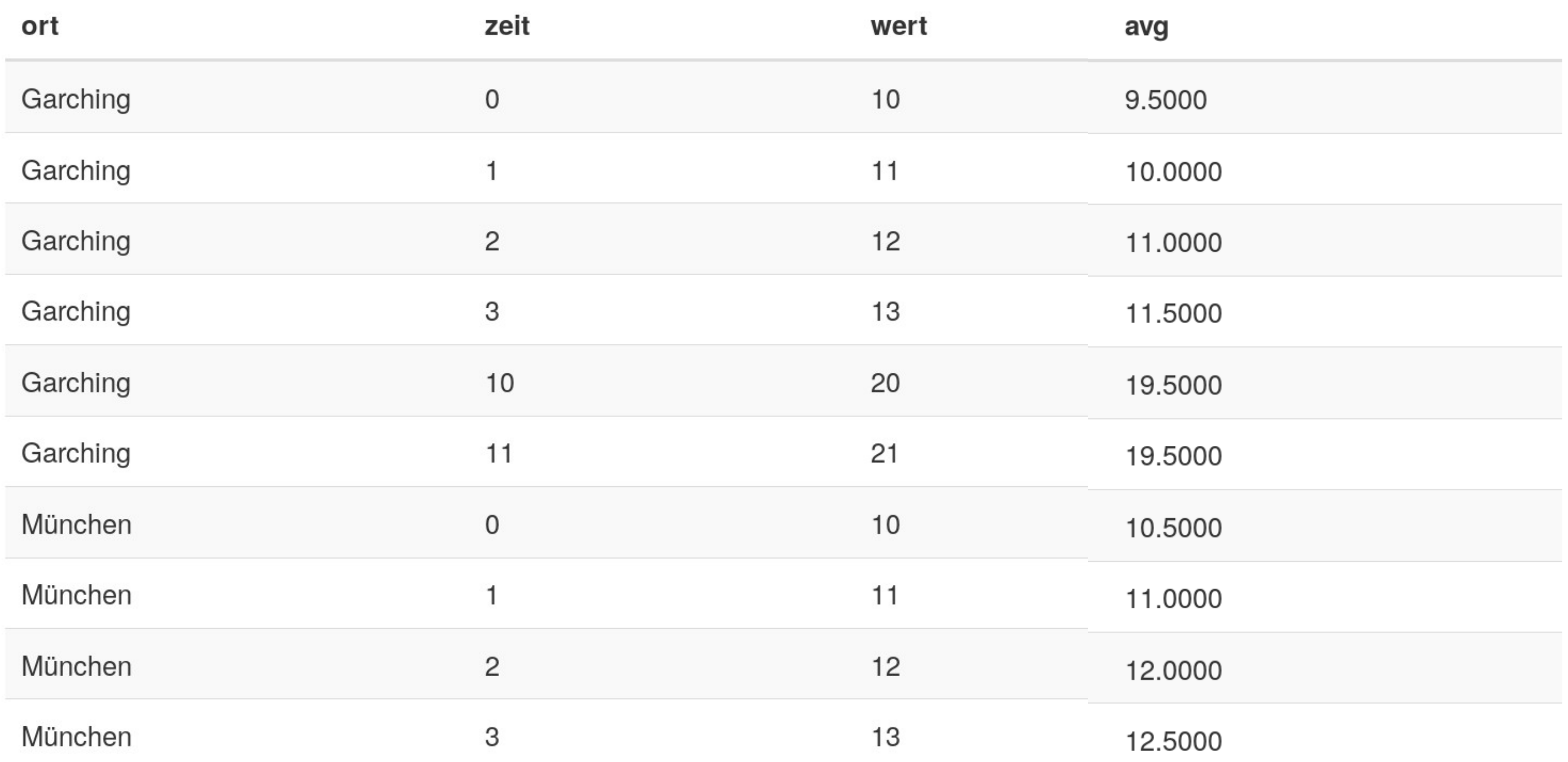
```
select *, avg(Wert) over (partition by ort
   order by zeit range between 1 preceding and 1 following)
from messungen order by Ort, Zeit
```



ort	zeit	wert	avg
Garching	0	10	9.5000
Garching	1	11	10.0000
Garching	2	12	11.0000
Garching	3	13	11.5000
Garching	10	20	19.5000
Garching	11	21	19.5000
München	0	10	10.5000
München	1	11	11.0000
München	2	12	12.0000
München	3	13	12.5000

Window-Functions: Explizit

```
select *, avg(Wert) over w from messungen order by Ort, Zeit
window w as (partition by ort
   order by zeit range between 1 preceding and 1 following)
```



ort	zeit	wert	avg
Garching	0	10	9.5000
Garching	1	11	10.0000
Garching	2	12	11.0000
Garching	3	13	11.5000
Garching	10	20	19.5000
Garching	11	21	19.5000
München	0	10	10.5000
München	1	11	11.0000
München	2	12	12.0000
München	3	13	12.5000

Window Funktionen in SQL

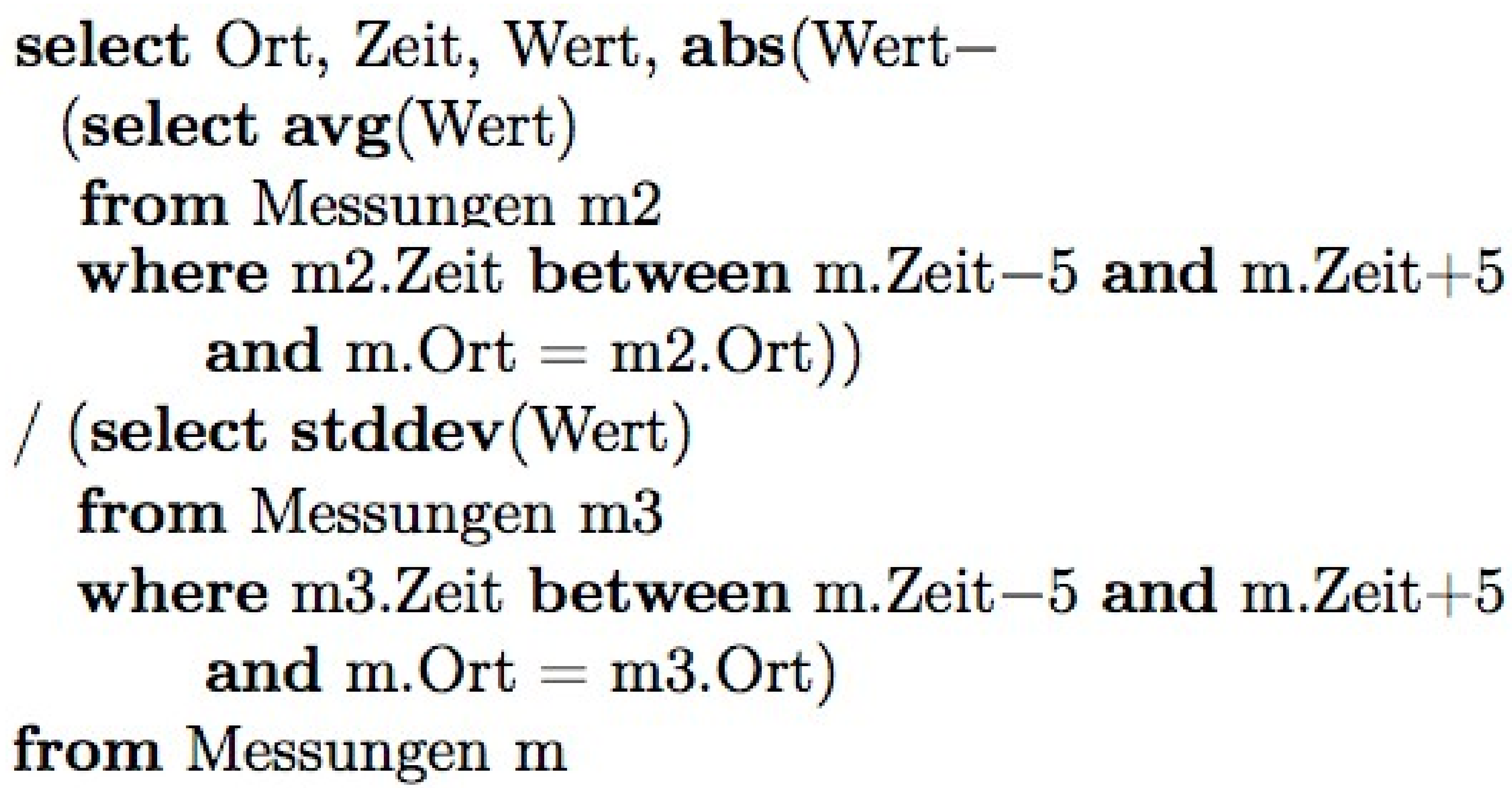
```
select Ort, Zeit, Wert, abs(Wert − (avg(Wert) over w)) / (stddev(Wert) over w)
from Messungen
window w as (
  partition by Ort
  order by Zeit
  range between 5 preceding and 5 following)
```

```
select avg(Wert) from Messungen m2
where m2.Zeit between m.Zeit−5 and m.Zeit+5 and m.ort=m2.Ort)
```

```
                            select stddev(Wert) from Messungen m3
     where m3.Zeit between m.Zeit−5 and m.Zeit+5 and m.ort=m3.Ort
```

Window-Functions ausdrückbar in SQL-92!

Komplexe Anfrage

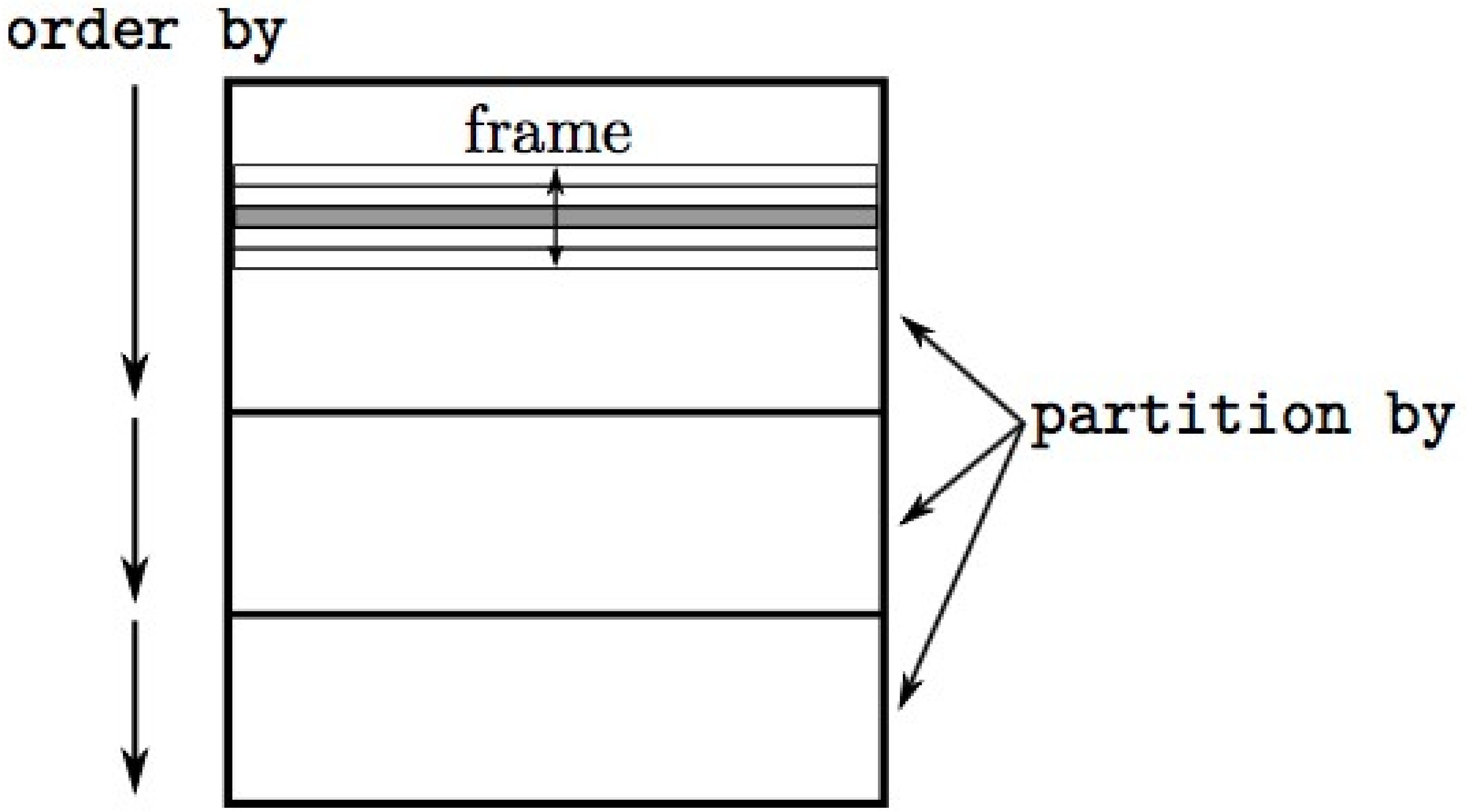


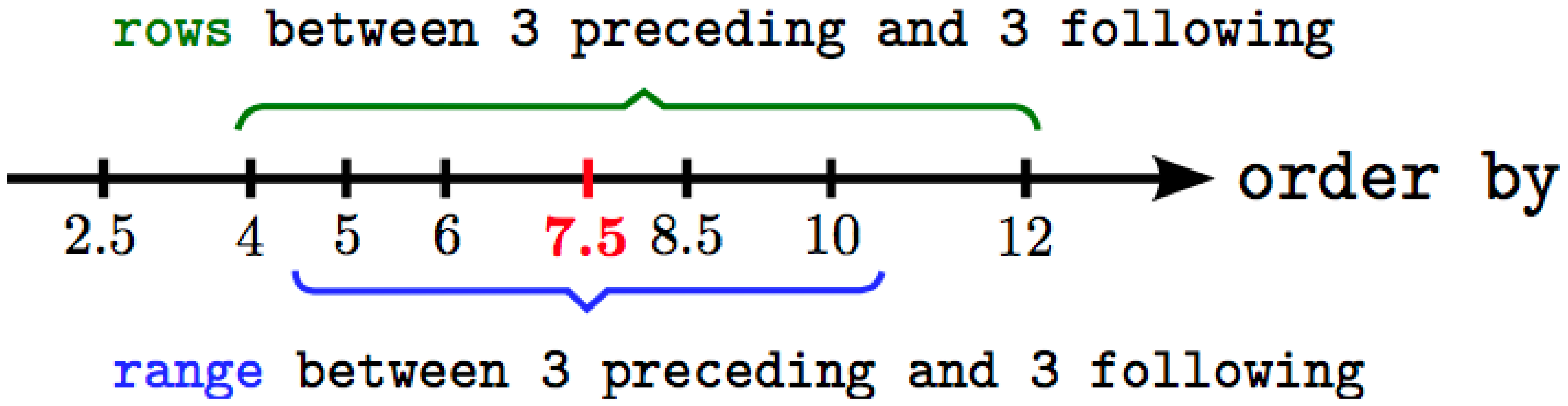
```
select Ort, Zeit, Wert, abs(Wert−
  (select avg(Wert)
   from Messungen m2
   where m2.Zeit between m.Zeit−5 and m.Zeit+5
         and m.Ort = m2.Ort))
/ (select stddev(Wert)
   from Messungen m3
   where m3.Zeit between m.Zeit−5 and m.Zeit+5
         and m.Ort = m3.Ort)
from Messungen m
```

Lag: Vorhergehendes Tupel im Frame

```
select Zeit, Wert,
       ((Wert − lag(Wert) over w) /
        (Zeit − lag(Zeit) over w)) as Änderungsrate
from Messungen
window w as (order by Zeit)
```

Zusammenhang: Frames / Partition / Sortierung

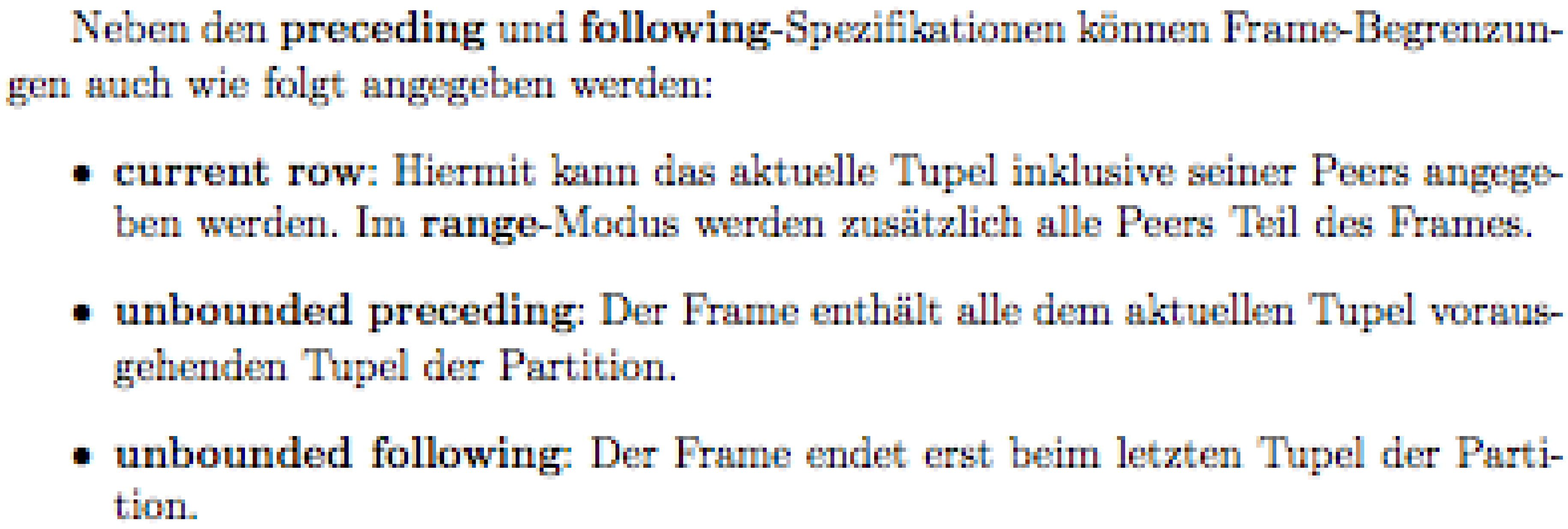



rows between 3 preceding and 3 following
order by
2.5
4
5
6
7.5
8.5
10
12
range between 3 preceding and 3 following

Medaillengewinner (in schön;-)

```
select Name, (case RangPlatz
                          when 1 then 'Gold'
                          when 2 then 'Silber'
                          else 'Bronze' end)
from (select Name, rank() over w as RangPlatz
        from Resultate
        window w as (order by Punkte desc))
where RangPlatz <= 3
```

Frame Begrenzungen

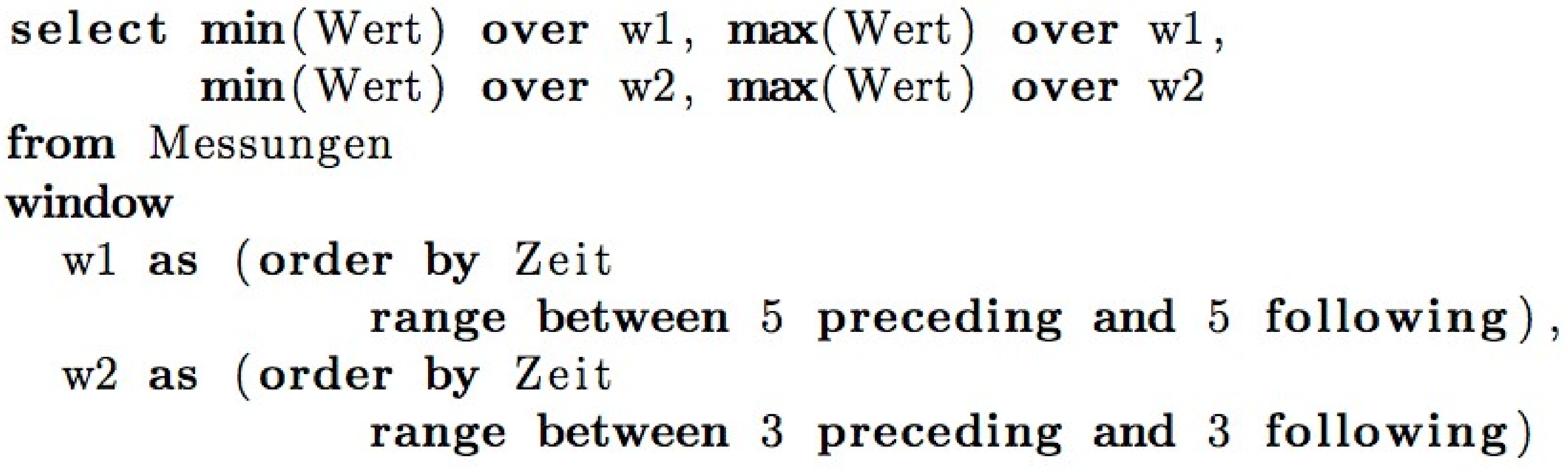


Neben den **preceding** und **following**-Spezifikationen können Frame-Begrenzungen auch wie folgt angegeben werden:

- **current row**: Hiermit kann das aktuelle Tupel inklusive seiner Peers angegeben werden. Im **range**-Modus werden zusätzlich alle Peers Teil des Frames.
- **unbounded preceding**: Der Frame enthält alle dem aktuellen Tupel vorausgehenden Tupel der Partition.
- **unbounded following**: Der Frame endet erst beim letzten Tupel der Partition.

```
select KundenID, Bestelldatum, sum(Preis) over
    (partition by KundenID,
                  extract(month from Bestelldatum),
                  extract(year from Bestelldatum)
     order by Bestelldatum
     range between unbounded preceding and current row)
from Verkäufe;
```

Explizite Windows-Klausel
Wiederverwendbarkeit



```
select min(Wert) over w1, max(Wert) over w1,
       min(Wert) over w2, max(Wert) over w2
from Messungen
window
  w1 as (order by Zeit
              range between 5 preceding and 5 following),
  w2 as (order by Zeit
              range between 3 preceding and 3 following)
```

Ranking innerhalb von Untergruppen

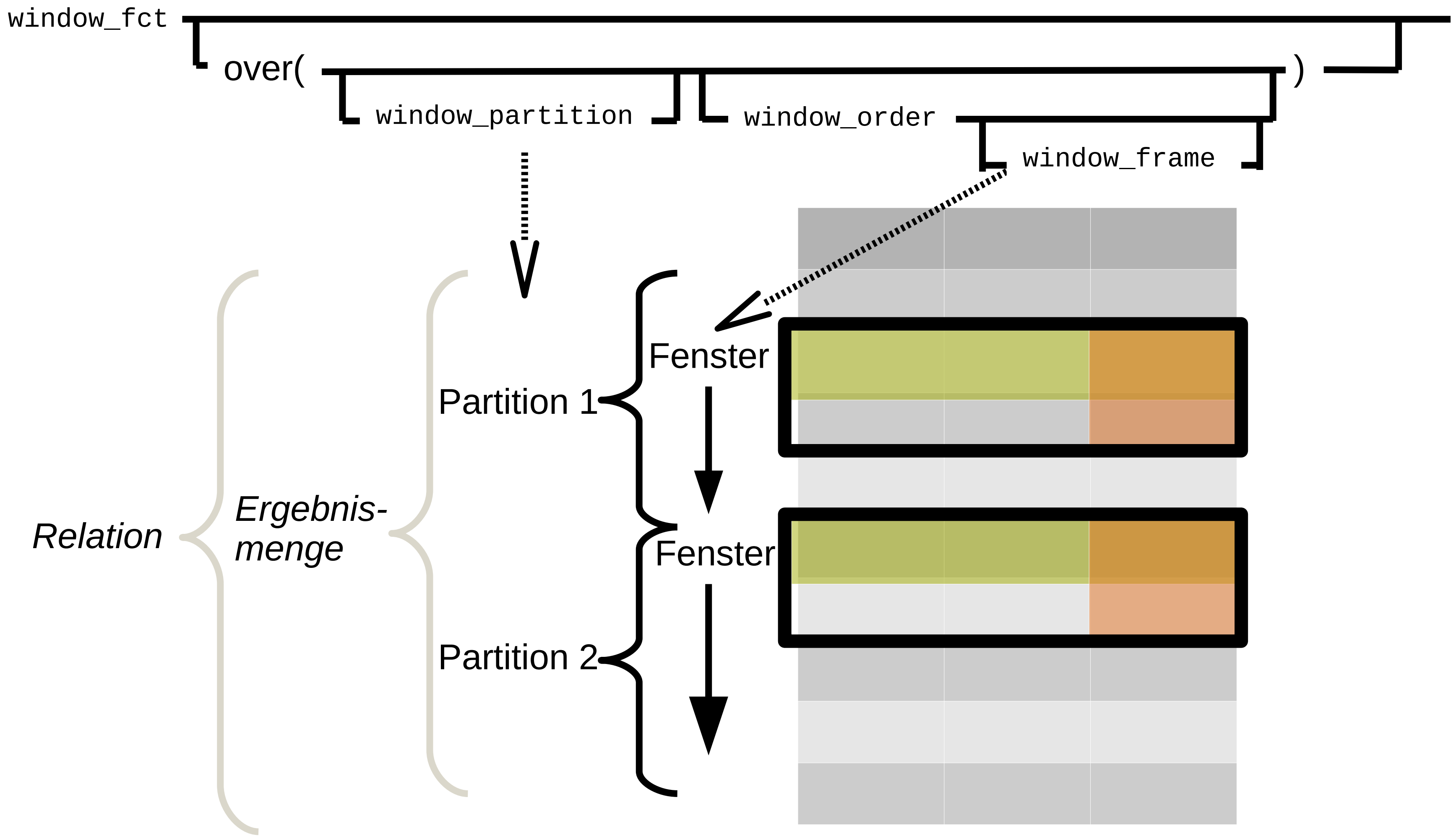
```
with KostenLageVergleich as (
   select m.Ort, m.Miete, k.Beitrag, l.Lage
   from Mietspiegel m, Kindergarten k, WohnLage l
   where m.Ort = k.Ort and k.Ort = l.Ort

select k.Ort, k.Lage, k.Miete + 3 * k.Beitrag as Kosten,
         rank() over (partition by k.Lage
                                   order by k.Miete + 3 * k.Beitrag asc) as LageRang
from KostenLageVergleich k
order by k.Lage, LageRang
```

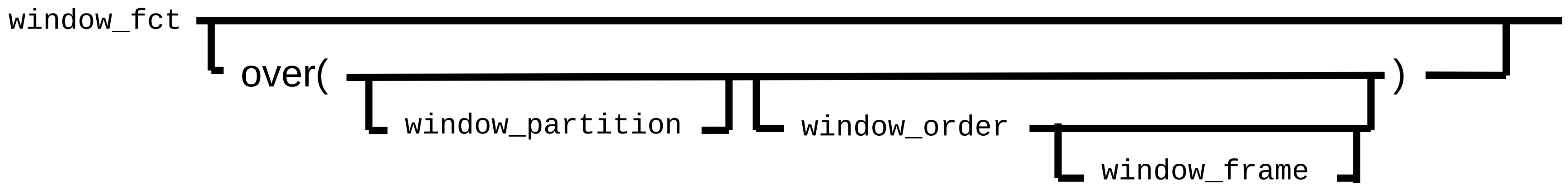
Die Partitionierung der (virtuellen) Relation *KostenLageVergleich* erfogt gemä des Attributs *Lage*. Als Ergebnis der Anfrage erhalten wir folgende Relation:

Ergebnis			
Ort	Lage	Kosten	LageRang
Bogenhausen	München-City	1900	1
Nymphenburg	München-City	2400	2
Unterföhring	München-Nord	1000	1
Ismaning	München-Nord	1500	2
Garching	München-Nord	1550	3
Grünwald	München-Süd	1400	1

Window-Functions (Übersicht)



Window-Functions (BNF)



```
<window_fct>            := <window_function_type> OVER <window_specification>

<window_function_type> := ROW_NUMBER() | RANK() | LEAD(<column>) | LAG(<column>) |
                          FIRST_VALUE(<column>) | LAST_VALUE(<column>) |
                          NTH_VALUE(<column>, <n>) | SUM(<column>) |
                          MIN(<column>) | MAX(<column>) | AVG(<column> | COUNT(<column>)

<window_specification> := [ <window_partition> ] [ <window_order> ] [ <window_frame> ]

<window_partition>     := PARTITION BY <column>

<window_order>         := ORDER BY <column>

<window_frame>         := [ROWS | GROUPS | RANGE ] BETWEEN
                          [ UNBOUNDED PRECEDING | <n> PRECEDING | CURRENT ROW ] AND
                          [ UNBOUNDED FOLLOWING | <n> FOLLOWING | CURRENT ROW ]
```