Datenbanken und SQL

(Woche 3 - Tag 4)

Agenda

Aggregatfunktionen in Verbindung mit RIGHT oder LEFT JOIN

- Motivation
- Beispiele
- ifNULL-Funktion
- Übungen

Aggregatfunktionen

in Verbindung mit RIGHT oder LEFT JOIN

Motivation

- Wir haben bisher etliche Aufgabenstellungen betrachtet, bei denen regelmäßig nur die aggregierten Werte **verknüpfbarer Entitäten** zu berechnen waren.
- Dies erschien sinnvoll, da wir uns bei der Vermittlung Aggregierter Funktionen zunächst auf dieses Themenfeld **konzentrieren** wollten.
- Wir wollen nun aber auch Aufgabestellungen betrachten, bei den wir Aggregierte Funktionen in Verbindung mit einem Einsatz von RIGHT und/oder LEFT JOIN verwenden, um auch **nicht-verknüpfbare Entitäten** berücksichtigen zu können.
- Auf diese Weise werden wir zum einen die **Einsetzbarkeit** aggregierter Funktionen **erweitern** und zum anderen eine weitere Motivation für das Themenfeld RIGHT und/oder LEFT JOIN benennen können.

Beispiel (1)

Wir wollen für alle Kunden von „Geld_her“ die Anzahl der von ihnen eingereichten Abrechnungen ermitteln.
Nun aber sollen auch Kunden berücksichtigt werden, die niemals die Seite von Geld_her besuchten, und daher **keiner Abrechnung** zugeordnet werden können.

Wir betrachten zunächst den bisherigen Lösungsansatz:

```
SELECT Kunde.Kunde_ID,Vorname,Nachname,COUNT(Abrechnung_ID)
FROM Kunde INNER JOIN Abrechnung ON Kunde.Kunde_ID=Abrechnung.Kunde_ID
GROUP BY Kunde.Kunde_ID,Vorname,Nachname
ORDER BY COUNT(Abrechnung_ID) DESC;
```

Kunde_ID	Vorname	Nachname	COUNT(Abrechnung_ID)
3	Witali	Myrnow	3
1	Elli	Rot	2
2	Vera	Deise	2
4	Rita	Myrnow	1
5	Eva	Hahn	1
7	Peter	Kaufnix	1

Der INNER JOIN konnte „Gala Nieda“ mit keiner Abrechnung verknüpfen, daher fehlt sie.
Dieses Problem kann durch den Einsatz von LEFT JOIN gelöst werden:

```
SELECT Kunde.Kunde_ID,Vorname,Nachname,COUNT(Abrechnung_ID)
FROM Kunde LEFT JOIN Abrechnung ON Kunde.Kunde_ID=Abrechnung.Kunde_ID
GROUP BY Kunde.Kunde_ID,Vorname,Nachname
ORDER BY COUNT(Abrechnung_ID) DESC;
```

Kunde_ID	Vorname	Nachname	COUNT(Abrechnung_ID)
3	Witali	Myrnow	3
1	Elli	Rot	2
2	Vera	Deise	2
4	Rita	Myrnow	1
5	Eva	Hahn	1
7	Peter	Kaufnix	1
6	Gala	Nieda	0

Beispiel (2)

Wir wollen die Gesamteinnahmen pro Produkttyp ermitteln.

Nun aber sollen auch Produkttypen berücksichtigt werden, die bisher noch nicht verkauft wurden und also **keinem Eintrag der Hilfstabelle zugeordnet** werden können.

Wir betrachten erneut zunächst den bisherigen Lösungsansatz:

```
SELECT Produkt.Produkt_id,Produkt_Name, SUM(Euro_Preis)
FROM Abrechnung_Produkt INNER JOIN Produkt ON Abrechnung_Produkt.Produkt_ID=Produkt.Produkt_id
GROUP BY Produkt.Produkt_id,Produkt_Name
ORDER BY SUM(Euro_Preis) DESC;
```

Produkt_id	Produkt_Name	SUM(Euro_Preis)
5	Do IT edition 1	392.00
3	solver 1000	126.76
2	tool 3.1	113.75
4	solver premium	90.10
1	tool 2.0	79.90

Der INNER JOIN konnte dem Produkt „TroppoCaro“ mit keinem Eintrag der Hilfstabelle verknüpfen, daher fehlt dieses.
Dieses Problem kann aber auch durch den „bloßen“ Einsatz von RIGHT JOIN (noch) **nicht gelöst** werden:

```
SELECT Produkt.Produkt_id,Produkt_Name, SUM(Euro_Preis)
FROM Abrechnung_Produkt RIGHT JOIN Produkt ON Abrechnung_Produkt.Produkt_ID=Produkt.Produkt_id
GROUP BY Produkt.Produkt_id,Produkt_Name
ORDER BY SUM(Euro_Preis) DESC;
```

Produkt_id	Produkt_Name	SUM(Euro_Preis)
6	TroppoCaro	1000.00
5	Do IT edition 1	392.00
3	solver 1000	126.76
2	tool 3.1	113.75
4	solver premium	90.10
1	tool 2.0	79.90

TroppoCaro wurde **keinmal** verkauft, dennoch Summe „1000“ !?

Begründung: Durch den RIGHT JOIN wird die Ausgabe von TroppoCaro erzwungen, zusammen mit dem Einzelpreis 1000.

Beispiel (2) - Lösung

Wir umgehen das Problem, indem wir zwar weiterhin die Ausgabe des Produktes „TroppoCaro“ mittels RIGHT JOIN „erzwingen“ (obwohl es mit keinem Eintrag der Hilfstabelle verknüpfbar ist), nun aber pro Produkt nicht die Summe der Einzelpreise ermitteln, sondern zunächst die Anzahl der Einkäufe berechnen lassen, um diese dann mit dem Einzelpreis zu multiplizieren:

```
SELECT Produkt.Produkt_id,
       Produkt_Name,
       COUNT(Abrechnung_Produkt.Produkt_ID)*Euro_Preis AS "Gesamteinnahme"
FROM Abrechnung_Produkt RIGHT JOIN Produkt ON Abrechnung_Produkt.Produkt_ID=Produkt.Produkt_id
GROUP BY Produkt.Produkt_id,Produkt_Name
ORDER BY COUNT(Abrechnung_Produkt.Produkt_ID)*Euro_Preis DESC;
```

Produkt_id	Produkt_Name	Gesamteinnahme
5	Do IT edition 1	392.00
3	solver 1000	126.76
2	tool 3.1	113.75
4	solver premium	90.10
1	tool 2.0	79.90
6	TroppoCaro	0.00

Beispiel (3)

Wir wollen nun **PRO Kunden:** ID, Vornamen, Nachnamen und die jeweilige Gesamtbestellsumme ausgeben lassen.
Da wir aber auch Kunden berücksichtigen wollen, die bisher noch nichts bestellt haben, arbeiten wir mit dem LEFT JOIN:

```
SELECT Kunde.Kunde_ID,Vorname,Nachname,SUM(Euro_Preis)
FROM Kunde LEFT JOIN Abrechnung ON Kunde.Kunde_ID=Abrechnung.Kunde_id
    LEFT JOIN Abrechnung_Produkt ON Abrechnung.Abrechnung_ID=Abrechnung_Produkt.Abrechnung_ID
    LEFT JOIN Produkt ON Abrechnung_Produkt.Produkt_ID=Produkt.Produkt_id
GROUP BY Kunde.Kunde_ID,Vorname,Nachname
ORDER BY SUM(Euro_Preis) DESC;
```

Kunde_ID	Vorname	Nachname	SUM(Euro_Preis)
1	Elli	Rot	226.83
3	Witali	Myrnow	206.88
2	Vera	Deise	177.63
4	Rita	Myrnow	152.44
5	Eva	Hahn	38.73
6	Gala	Nieda	*NULL*
7	Peter	Kaufnix	*NULL*

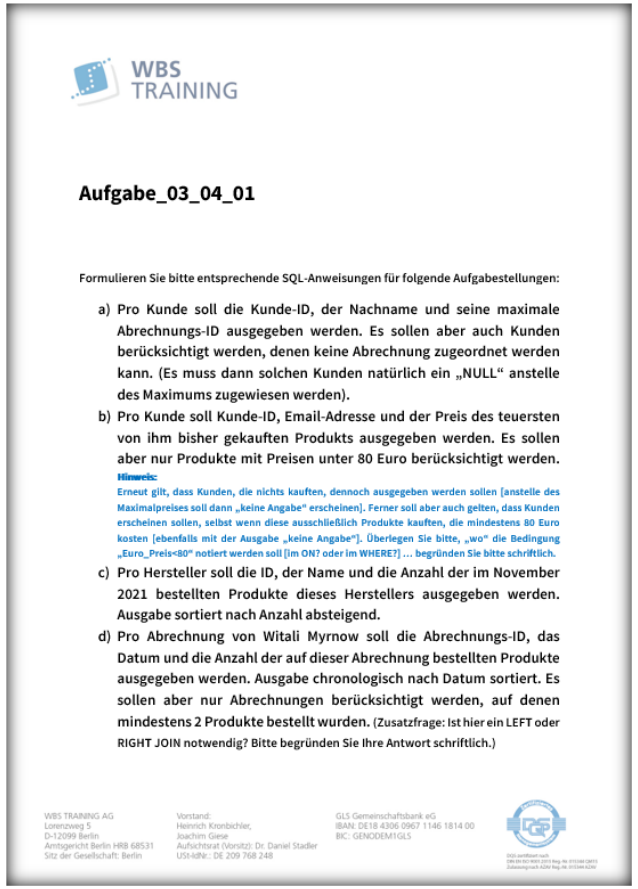
Nur die Aggregatfunktion **COUNT** gibt bei fehlenden Verknüpfungen den Wert **„0“** zurück.
Alle anderen haben in solchen Fällen den Rückgabewert **NULL**.

Beispiel (3) – „aufgehübscht“

```
SELECT Kunde.Kunde_ID,Vorname,Nachname, ifNULL(SUM(Euro_Preis),0)
FROM Kunde LEFT JOIN Abrechnung ON Kunde.Kunde_ID=Abrechnung.Kunde_id
    LEFT JOIN Abrechnung_Produkt ON Abrechnung.Abrechnung_ID=Abrechnung_Produkt.Abrechnung_ID
    LEFT JOIN Produkt ON Abrechnung_Produkt.Produkt_ID=Produkt.Produkt_id
GROUP BY Kunde.Kunde_ID,Vorname,Nachname
ORDER BY SUM(Euro_Preis) DESC;
```

Kunde_ID	Vorname	Nachname	ifNULL(SUM(Euro_Preis),0)
1	Elli	Rot	226.83
3	Witali	Myrnow	206.88
2	Vera	Deise	177.63
4	Rita	Myrnow	152.44
5	Eva	Hahn	38.73
6	Gala	Nieda	0.00
7	Peter	Kaufnix	0.00

Gemeinsame Übung („Live-Coding“) -> A_03_04_01



WBS TRAINING

Aufgabe_03_04_01

Formulieren Sie bitte entsprechende SQL-Anweisungen für folgende Aufgabestellungen:

a) Pro Kunde soll die Kunde-ID, der Nachname und seine maximale Abrechnungs-ID ausgegeben werden. Es sollen aber auch Kunden berücksichtigt werden, denen keine Abrechnung zugeordnet werden kann. (Es muss dann solchen Kunden natürlich ein „NULL“ anstelle des Maximums zugewiesen werden).

b) Pro Kunde soll Kunde-ID, Email-Adresse und der Preis des teuersten von ihm bisher gekauften Produkts ausgegeben werden. Es sollen aber nur Produkte mit Preisen unter 80 Euro berücksichtigt werden.
Hinweis:
Erneut gilt, dass Kunden, die nichts kauften, dennoch ausgegeben werden sollen [anstelle des Maximalpreises soll dann „keine Angabe“ erscheinen]. Ferner soll aber auch gelten, dass Kunden erscheinen sollen, selbst wenn diese ausschließlich Produkte kauften, die mindestens 80 Euro kosten [ebenfalls mit der Ausgabe „keine Angabe“]. Überlegen Sie bitte, „wo“ die Bedingung „Euro_Preis<80“ notiert werden soll [im ON? oder im WHERE?] … begründen Sie bitte schriftlich.

c) Pro Hersteller soll die ID, der Name und die Anzahl der im November 2021 bestellten Produkte dieses Herstellers ausgegeben werden. Ausgabe sortiert nach Anzahl absteigend.

d) Pro Abrechnung von Witali Myrnow soll die Abrechnungs-ID, das Datum und die Anzahl der auf dieser Abrechnung bestellten Produkte ausgegeben werden. Ausgabe chronologisch nach Datum sortiert. Es sollen aber nur Abrechnungen berücksichtigt werden, auf denen mindestens 2 Produkte bestellt wurden. (Zusatzfrage: Ist hier ein LEFT oder RIGHT JOIN notwendig? Bitte begründen Sie Ihre Antwort schriftlich.)

Vielen Dank für Ihre Aufmerksamkeit!