Datenbanken und SQL



(Woche 3 - Tag 3)



Agenda

Aggregatfunktionen

- Definition + Motivation
- Ausgabe nicht-gruppierter Einzelwerte
 - > MIN
 - > MAX
 - ➤ COUNT
 - > SUM
 - AVG
 - Stolperfallen + "schlechter Stil"



Aggregatfunktionen



Definition + Motivation

- "Aggregieren" bedeutet "anhäufen" oder auch "zusammenfügen". Genau dies leisten Aggregatfunktionen, die Werte (z.B. zu einer Summe) zusammenfassen.
- Im Gegensatz zu den meisten bisherigen SQL-Befehlen, bei denen eine Schleife gestartet wurde, um (unter Berücksichtigung etwaiger Bedingungen) die jeweils verlangte Funktionalität PRO Datensatz auszuführen, kann die Ausgabe bei Aggregatfunktion nicht für jeden einzelnen Datensatz geschehen, sondern gelingt erst NACH Durchsicht aller (anzusprechenden) Datensätze.
- Aggregatfunktionen sind mathematische Funktionen. Damit ist die Motivation für deren Einsatz aber offensichtlich, denn natürlich ist eine **mathematische Analyse** der in einer Datenbank gelisteten (Zahlen)-Werte oft von großer Bedeutung.
- Aggregatfunktionen können in Einzelfällen auch auf Nicht-Zahlen Typen angewendet werden. Möglichkeiten und Grenzen werden wir im Folgenden betrachten.



Nicht-gruppierte Einzelwerte



Aggregatfunktion MIN(...)

(Nicht-gruppierte) Aggregatfunktionen beziehen sich auf alle Datensätze, die <u>ohne</u> Aggregatfunktion ausgegeben worden wären. Hierzu ein erstes Beispiel an Hand der MIN-Funktion, die (natürlich) das **Minimum**

ermittelt:

SELECT Kunde_ID **FROM** Kunde;

Dies sind die Datensätze, die (in diesem Fall) ausgegeben werden, wenn man auf den Einsatz einer Aggregatfunktion **verzichtet**.



Kunde_ID

Dies ist das Minimum aller (oben dargestellten) Werte

SELECT MIN(Kunde_ID) FROM Kunde;

Falls einem die Überschrift nicht gefällt: SELECT MIN(Kunde_ID) AS "Kleinste Kunden-ID" FROM Kunde;

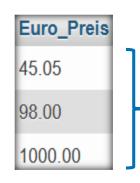




Aggregatfunktion MIN(...)

Für das Ergebnis einer Aggregatfunktion ist es (wie bei jeder mathematischen Funktion) von zentraler Bedeutung, auf welche Werte sich die jeweilige Aggregatfunktion bezieht. Daher ein weiteres Beispiel dazu:

SELECT Euro_Preis **FROM** Produkt **WHERE** Produkt_ID > 3;



Dies sind die Datensätze, die (in diesem Fall) ausgegeben werden, wenn man auf den Einsatz einer Aggregatfunktion **verzichtet**.

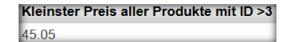
SELECT MIN(Euro_Preis) FROM Produkt WHERE Produkt_ID > 3;



Dies ist das Minimum aller (oben dargestellten) Werte

Keine Angst vor langen Überschriften ;-)

SELECT MIN(Euro_Preis) **AS** "Kleinster Preis aller Produkte mit ID > 3" **FROM** Produkt **WHERE** Produkt ID > 3;





MIN(...) - andere Typen

Wie schon bei ORDER BY können wir auch bei der Aggregatfunktion MIN vom "kleinsten" Datum (=ältestes Datum) oder auch von einem "kleinsten" Text (alphabetisch betrachtet "der erste") sprechen:

SELECT MIN(Datum) FROM Abrechnung;

MIN(Datum) 2021-05-05

SELECT MIN(Hersteller_Name) FROM Hersteller;

MIN(Hersteller_Name)
AntiByte



Aggregatfunktion MAX(...)

Die Aggregatfunktion MAX ermittelt (wie zu erwarten ©) das Maximum aller betrachteten Werte. Es gelten die entsprechenden Aussagen, die wir bereits bei der Funktion MIN kennenlernten. Wir betrachten 3 Beispiele:

SELECT MAX(Euro_Preis) **FROM** Produkt;

MAX(Euro_Preis) 1000.00

SELECT MAX(Datum) FROM Abrechnung;

MAX(Datum) 2022-02-14

SELECT MAX(Hersteller_Name) **FROM** Hersteller;

MAX(Hersteller_Name)
UltraBug

Bemerkung:



Aggregatfunktion MAX(...)

Die Aggregatfunktion MAX ermittelt (wie zu erwarten ©) das Maximum aller betrachteten Werte. Es gelten die entsprechenden Aussagen, die wir bereits bei der Funktion MIN kennenlernten. Wir betrachten 3 Beispiele:

SELECT MAX(Euro_Preis) **FROM** Produkt;

MAX(Euro_Preis) 1000.00

SELECT MAX(Datum) FROM Abrechnung;

MAX(Datum) 2022-02-14

SELECT MAX(Hersteller_Name) **FROM** Hersteller;



Bemerkung:

Die MIN- und MAX-Funktionen können vordergründig durch ORDER BY ... [DESC] LIMIT 1 ersetzt werden. Folgendes Beispiel zeigt aber, dass sie auch eine eigenständige Berechtigung haben:



Aggregatfunktion MAX(...)

Die Aggregatfunktion MAX ermittelt (wie zu erwarten ©) das Maximum aller betrachteten Werte. Es gelten die entsprechenden Aussagen, die wir bereits bei der Funktion MIN kennenlernten. Wir betrachten 3 Beispiele:

SELECT MAX(Euro_Preis) **FROM** Produkt;

MAX(Euro_Preis) 1000.00

SELECT MAX(Datum) FROM Abrechnung;

MAX(Datum) 2022-02-14

SELECT MAX(Hersteller_Name) FROM Hersteller;

MAX(Hersteller_Name)
UltraBug

Bemerkung:

Die MIN- und MAX-Funktionen können vordergründig durch ORDER BY ... [DESC] LIMIT 1 ersetzt werden. Folgendes Beispiel zeigt aber, dass sie auch eine eigenständige Berechtigung haben:

SELECT MIN(Datum), MAX(Datum) FROM Abrechnung;

MIN(Datum) MAX(Datum)
2021-05-05 2022-02-14



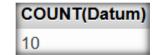
Aggregatfunktion COUNT(...)

Die Aggregatfunktion COUNT(Attribut) zählt alle Einträge des Attributs, deren Wert ungleich NULL ist.

SELECT COUNT(Abrechnung_ID) FROM Abrechnung;



SELECT COUNT(Datum) FROM Abrechnung;



Es erscheint das identische Ergebnis, da **kein** Datensatz in Abrechnung den Kalenderdatums-Wert **NULL** hat.

SELECT COUNT(*) FROM Abrechnung;



Alternativ kann man auch die Anzahl der Datensätze zählen.

Mit COUNT(DISTINCT ...) kann die Anzahl der <u>unterschiedlichen</u> Einträge (ungleich NULL) ermittelt werden.



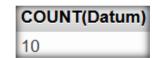
Aggregatfunktion COUNT(...)

Die Aggregatfunktion COUNT(Attribut) zählt alle Einträge des Attributs, deren Wert ungleich NULL ist.

SELECT COUNT(Abrechnung_ID) FROM Abrechnung;

COUNT(Abrechnung_ID)
10

SELECT COUNT(Datum) FROM Abrechnung;



Es erscheint das identische Ergebnis, da **kein** Datensatz in Abrechnung den Kalenderdatums-Wert **NULL** hat.

SELECT COUNT(*) FROM Abrechnung;



Alternativ kann man auch die Anzahl der Datensätze zählen.

Mit COUNT(DISTINCT ...) kann die Anzahl der unterschiedlichen Einträge (ungleich NULL) ermittelt werden.

Beispiel:

SELECT COUNT(Nachname), COUNT(DISTINCT Nachname) FROM Kund





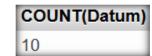
Aggregatfunktion COUNT(...)

Die Aggregatfunktion COUNT(Attribut) zählt alle Einträge des Attributs, deren Wert ungleich NULL ist.

SELECT COUNT(Abrechnung_ID) FROM Abrechnung;

COUNT(Abrechnung_ID)
10

SELECT COUNT(Datum) FROM Abrechnung;



Es erscheint das identische Ergebnis, da **kein** Datensatz in Abrechnung den Kalenderdatums-Wert **NULL** hat.

SELECT COUNT(*) FROM Abrechnung;



Alternativ kann man auch die Anzahl der Datensätze zählen.

Mit COUNT(DISTINCT ...) kann die Anzahl der unterschiedlichen Einträge (ungleich NULL) ermittelt werden.

Beispiel:

SELECT COUNT(Nachname), COUNT(DISTINCT Nachname) FROM Kunc



(Es gibt 7 Kundennachnamen ungleich NULL, aber nur 6 unterschiedliche, da es zwei Kunden mit Nachnamen "Myrnow" gibt).



Die Aggregatfunktionen SUM und AVG ermitteln Summe und Durchschnitt (Average) der betrachteten Werte.

Durchschnittspreis aller von "Geld_her" angeboten Waren:

SELECT AVG(Euro_Preis) FROM Produkt;

AVG(Euro_Preis) 202.245000

Gesamtbestellsume (aller bisher bei "Geld_her" gekauften Waren):

SELECT SUM(Euro_Preis) **FROM** Abrechnung_Produkt, Produkt **WHERE** Abrechnung_Produkt.Produkt_ID=Produkt.Produkt_ID;

SUM(Euro_Preis) 802.51



Die Aggregatfunktionen SUM und AVG ermitteln Summe und Durchschnitt (Average) der betrachteten Werte.

Durchschnittspreis aller von "Geld_her" angeboten Waren:

SELECT AVG(Euro_Preis) FROM Produkt;

AVG(Euro_Preis) 202.245000

Gesamtbestellsume (aller bisher bei "Geld_her" gekauften Waren):

SELECT SUM(Euro_Preis) **FROM** Abrechnung_Produkt, Produkt **WHERE** Abrechnung_Produkt.Produkt_ID=Produkt.Produkt_ID;

SUM(Euro_Preis) 802.51

Der Einsatz von SUM oder AVG auf Attribute vom Typ DATE oder VARCHAR ist sinnlos, führt aber nicht zu einer Fehlermeldung:



Die Aggregatfunktionen SUM und AVG ermitteln Summe und Durchschnitt (Average) der betrachteten Werte.

Durchschnittspreis aller von "Geld_her" angeboten Waren:

SELECT AVG(Euro_Preis) FROM Produkt;



Gesamtbestellsume (aller bisher bei "Geld_her" gekauften Waren):

SELECT SUM(Euro_Preis) **FROM** Abrechnung_Produkt, Produkt **WHERE** Abrechnung_Produkt.Produkt_ID=Produkt.Produkt_ID;



Der Einsatz von SUM oder AVG auf Attribute vom Typ DATE oder VARCHAR ist sinnlos, führt aber nicht zu einer Fehlermeldung:

SELECT SUM(Datum), AVG(Datum) FROM Abrechnung;

SUM(Datum)	AVG(Datum)
202119212	20211921.2000



Die Aggregatfunktionen SUM und AVG ermitteln Summe und Durchschnitt (Average) der betrachteten Werte.

Durchschnittspreis aller von "Geld_her" angeboten Waren:

SELECT AVG(Euro_Preis) FROM Produkt;



Gesamtbestellsume (aller bisher bei "Geld_her" gekauften Waren):

SELECT SUM(Euro_Preis) **FROM** Abrechnung_Produkt, Produkt **WHERE** Abrechnung_Produkt.Produkt_ID=Produkt.Produkt_ID;



Der Einsatz von SUM oder AVG auf Attribute vom Typ DATE oder VARCHAR ist sinnlos, führt aber nicht zu einer Fehlermeldung:

SELECT SUM(Datum), AVG(Datum) FROM Abrechnung;

SUM(Datum) AVG(Datum) 202119212 20211921.2000

SELECT SUM(Nachname), AVG(Nachname) FROM Kund

SUM(Nachname) AVG(Nachname)
0 0

