SQL

nach KonzMod_2011SS_VL08_SQL.pdf

Aggregationsfunktionen

min(<Attribut>)	Minimaler existierender Attributwert
max(<Attribut>)	Maximaler existierender Attributwert
count([distinct]<Attribut>)	Anzahl der [untersch.] existierienden Attributwerte (= Anzahl der “Zeilen”)
avg([distinct]<Attribut>)	Durchschn. der [untersch.] Attributwerte
sum([distinct]<Attribut>)	Summe der [untersch.] Attributwerte

Ausführreihenfolge

1. FROM
2. WHERE
3. GROUP BY
4. HAVING
5. SELECT
6. ORDER BY

Beispiele

users

user_id	**name**
1	chris
2	moe
3	steffi
4	phil
5	pasi

ages

name	**age**
chris	24
moe	17
steffi	18
pasi	3
franz	12

Zähle die User

SELECT count(*) **AS** count_users **FROM** users;

count_users
5

Hole die höchste ID

SELECT max(user_id) **AS** newest_uid **FROM** users;

newest_uid
5

MERKE !
Die höchte user_id muss nicht gleich der Anzahl der user sein
(zB. chris löscht seinen Account: newest_uid = 5; count_users = 4)

Hole das Alter des Users mit der ID 1

SELECT age **FROM** users,ages **WHERE** users.user_id = 1 **AND** users.name = ages.name;

age
24

Hole die IDs der User die nicht in der Tabelle "ages" vorkommen

SELECT DISTINCT user_id **AS** user_without_age **FROM** users,ages **WHERE** users.name **NOT IN** (**SELECT** name **FROM** ages);

user_without_age
4

MERKE !
Da "FROM users,ages" das Kreuzprodukt der 2 Tabellen liefert müssen wir ein DISTINCT verwenden, da die ID 4 mehrfach ausgegeben werden würde !

Hole alle Infos über User die sowohl Alter als auch User-ID besitzen, aufsteigend nach Alter sortiert

SELECT * FROM ages **NATURAL JOIN** users **ORDER BY** age **ASC**;

name	age	user_id
pasi	3	5
moe	17	2
steffi	18	3
chris	24	1

Hole Name und Rang der 2 ältesten User, die eine UserID besitzen, dem Rang nach aufsteigend sortiert

Erstelle eine View die uns das aus der Augabe vorher "gibt"
CREATE VIEW user_n_age **AS SELECT** * **FROM** ages **NATURAL JOIN** users **ORDER BY** age **ASC**;

Hole die 2 ältesten User unter Verwendung dieser View (kein LIMIT o.ä.)
SELECT count(*) AS Rang,ua1.name
FROM user_n_age ua1, user_n_age ua2
WHERE ua1.age <= ua2.age
GROUP BY ua1.name
HAVING count(*) <= 2
ORDER BY count(*) **ASC**;

Lösche die oben erstellte View
DROP VIEW user_n_age;

Rang	Name
1	chris
2	steffi

Für Genaueres die einschlägige Literatur / Bekannte quälen ;)