

:۱

الف : پارتیشن بندی یک فرآیند برای پایگاه داده است که جداول بسیار بزرگ به چندین قسمت کوچکتر تقسیم می شوند. با تقسیم یک جدول بزرگ به جداول کوچکتر و جداگانه، قسمت هایی که فقط به بخشی از داده دسترسی دارند سپس سریعتر اجرا می شوند، زیرا داده های کمتری برای اسکن وجود دارد.

### مزایای Partitioning:

- بهبود عملکرد برای queryها، به ویژه queryهایی که بر روی داده های اخیر انجام می شوند
- سهولت مدیریت و نگهداری: امکان انجام عملیات مثل Rebuild Index بر روی یک Partition به جای Table کامل
- مقیاس پذیری بهتر: امکان اضافه کردن Partition جدید و توزیع بار در کلاسترهای مختلف
- افزایش امنیت: پارتیشن بندی دیسک می تواند به افزایش امنیت داده های شما کمک کند. به عنوان مثال، می توانید یک پارتیشن جداگانه برای ذخیره داده های حساس خود ایجاد کنید و دسترسی به آن را محدود کنید.

### انواع Partitioning:

- Range Partitioning: بر اساس محدوده ی مقادیر یک ستون
  - List Partitioning: بر اساس مقادیر خاص یک ستون
  - Hash Partitioning: بر اساس خروجی یک تابع Hash
- مراحل پیاده سازی:

1. تعیین معیار Partitioning: ستون هایی که قرار است برای Partitioning استفاده شوند، تعیین می کنیم.
2. انتخاب روش Partitioning: روش هایی مثل Range Partitioning، Hash Partitioning، List Partitioning وجود دارد.
3. ایجاد تابع Partition و تعریف Partitionها: تابعی برای تعیین اینکه هر رکورد متعلق به کدام Partition باشد، نوشته می شود.
4. اعمال Partitioning بر روی جدول: دستور alter table را اجرا می کنیم تا Partitioning بر روی جدول اعمال شود

1ب: زیرا برا هر زمان بايد جزييات آن زمان به همراه سال و ماه و ... بايد تعريف بشود يعنى تمام كليد اصلى آن بايد شامل تمام اين ها باشد در غير اينصورت يكتا نميشود كه اگر اين كليد خارجى باشد يعنى براى هر فيلد جدول زمان يك فيلد در جدول اصلى داريم كه دوباره كارى است و جدولى بيهوده ساخته ايم همچنين مان تشكيل كلاس ها به عنوان يك ويژگى زمانى، باعث تغييرات مكرر و پويا در ساختار پايگاه داده مى شود. اين تغييرات ممكن است منجر به خرابى پايگاه داده شود يا اطلاعات را به خطر بياندازد

ا ج: تراكنش پايگاه داده مجموعه اى از پرس وجوها و عبارات بروررسانى است. استاندارد SQL بيان مى كند كه وقتى يك عبارت SQL اجرا مى شود، يك تراكنش به طور ضمنى شروع مى شود. يكي از عبارات SQL زير بايد تراكنش را پايان دهد:

**Commit work:** تراكنش فعلى را انجام مى دهد، به اين معنى كه بهروررسانى هاى انجام شده توسط تراكنش در پايگاه داده دائمى مى شوند. پس از انجام تراكنش، يك تراكنش جديد به طور خودكار شروع مى شود.

**Rollback work:** اين دستور باعث مى شود تراكنش فعلى به حالت اوليه خود برگردد. اين بدان معناست كه تمام بهروررسانى هاى انجام شده توسط عبارات SQL در تراكنش به حالت قبل از اجراى تراكنش بازمى گردند. بنابراين، وضعيت پايگاه داده به حالتى كه قبل از اجراى تراكنش بود بازمى گردد.

عقب گرد تراكنش در شرايطى كه خطايى در حين اجراى تراكنش شناسايى شود، مفيد است. دستور commit از جهاتى شبیه زمانى است كه تغييرات انجام شده در يك سند در حال ويرايش را ذخيره مى كنيم، در حالى كه عقب گرد مشابه ترك عمليات ويرايش بدون ذخيره تغييرات در سند است. هنگامى كه يك تراكنش تثبيت ميشود نميتوان با انجام عملگر rollback نتايج تراكنش را به حالت قبل از اجرا بازگرداند. سيستم پايگاه داده تضمين ميكند كه در صورت وقوع برخى شكست ها مانند خطا در يكي از عبارات SQL قطع برق و يا خرابى سيستم اگر هنوز تراكنش دستور commit را اجرا نكرده باشد؛ نتايج تراكنش عقب گرد خواهد شد. در صورت وقوع قطع برق يا خرابى سيستم دستور عقب گرد زمانى اجرا ميشود كه سيستم مجدداً شروع به كار كند (restartشود).

به عنوان مثال ديگر برنامه دانشگاه را در نظر بگيريد فرض ميكنيم ويژگى toot\_cred از همه تايل در رابطه student به ازاي هر درسى كه دانشجو با موفقيت ميگذراند، به روز رسانی شود. برای این کار وقتی رابطه takes با گذراندن موفق یک درس توسط یک دانشجو به روز میشود با گرفتن یک نمره مناسب تايل متناظر در رابطه student باید به روز رسانی شود اما اگر برنامه ای که باید این دو به روز رسانی را انجام دهد بعد از اجرای اولین به روز رسانی و قبل از اجرای به روز رسانی دوم خراب شود داده های پايگاه داده ناسازگار ميشوند.

:2

الف:

Employee\_id: char

dept\_name: varchar

Name: varchar

Date\_of\_birth: date

Hire\_date: date

salary: numeric

Is\_manager: boolean

Address: varchar or text

Photo: blob

ب:

```
1 CREATE TABLE employee (  
2   Employee_id char(5) PRIMARY KEY,  
3   dept_name VARCHAR(100) NOT NULL,  
4   Name VARCHAR(50) NOT NULL,  
5   Date_of_birth DATE CHECK (YEAR(Date_of_birth) > 1963),  
6   Hire_date DATE NOT NULL,  
7   salary numeric(8,2) CHECK (salary BETWEEN 20000 AND 1000000),  
8   Is_manager BOOLEAN NOT NULL,  
9   Address VARCHAR(200) NOT NULL,  
10  Photo blob NOT NULL  
11 );
```

ج:

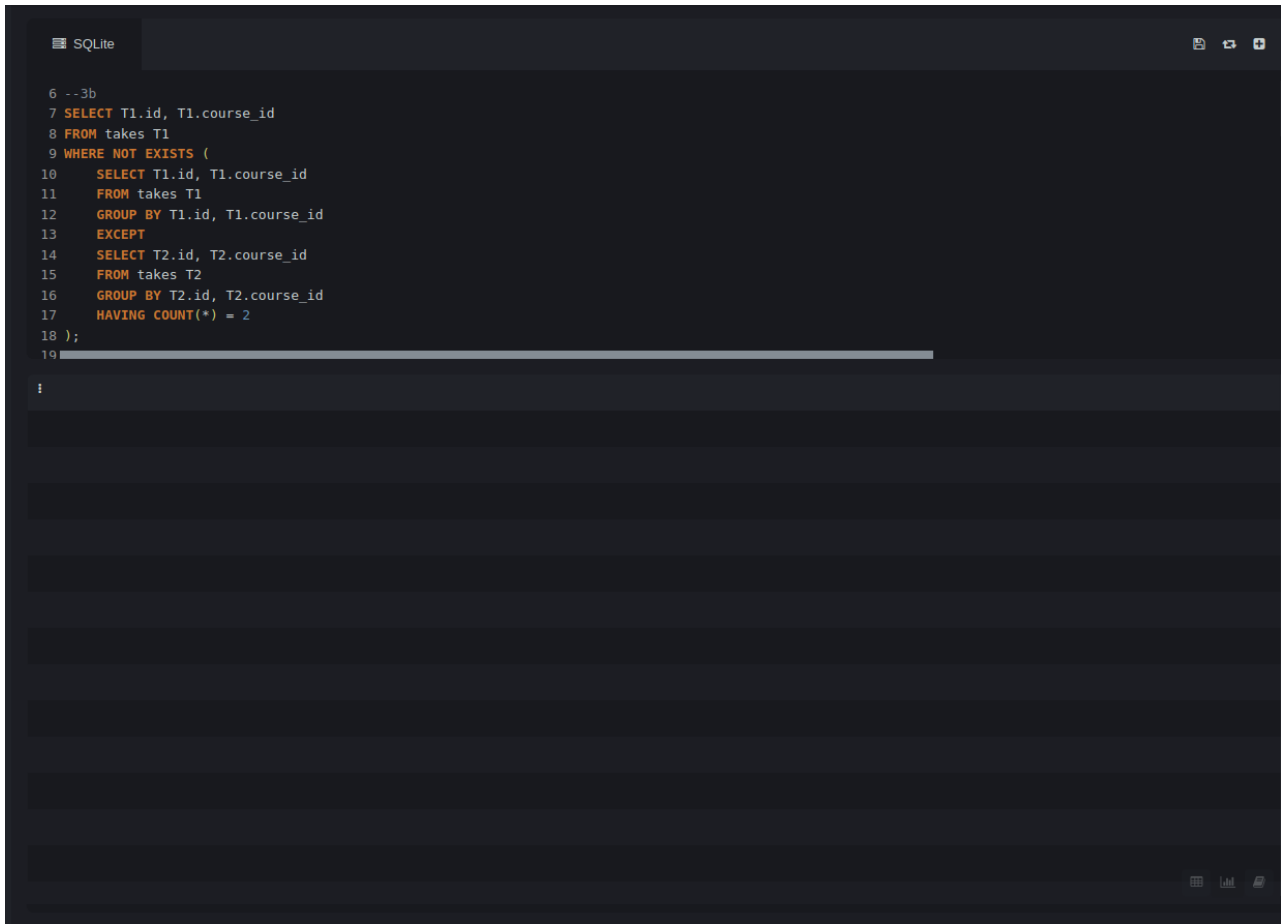
```
1 INSERT INTO employee (Employee_id, dept_name, Name, Date_of_birth, Hire_date, salary, Is_manager, Address)
2 VALUES (1, '1', '1', '1990-05-20', '2020-04-01', 50000, FALSE, '1');
3
4 INSERT INTO employee (Employee_id, dept_name, Name, Date_of_birth, Hire_date, salary, Is_manager, Address)
5 VALUES (2, '2', '2', '1995-11-30', '2022-09-15', 80000, TRUE, '2');
```

:3

:A

```
1 SELECT *
2 FROM student
3 WHERE name LIKE '%X%'
4
```

ID	name	dept_name	tot_cred
58413	Xiong	Athletics	27
72979	Guix	Astronomy	117
92040	Alexandri	Athletics	66
10880	Xue	Languages	94
59908	Cox	Civil Eng.	0
21766	Cox	Astronomy	74
40992	Xing	Psychology	93
69850	Alexandri	Finance	80
73165	Fox	Statistics	19
29002	Duxbury	History	29
78116	Xiao	Civil Eng.	65
62749	Giroux	Math	49
4034	Xie	Elec. Eng.	64
47670	Xue	Pol. Sci.	55
7149	Xin	Pol. Sci.	120
63502	Xie	Languages	69
31035	Arnoux	Civil Eng.	92
44304	Cox	English	31
14499	Axte	Biology	115

A screenshot of the SQLite IDE interface. The top bar shows the title 'SQLite' and three icons on the right. The main area contains a SQL query with line numbers 6 through 19. The query is a complex SELECT statement using a NOT EXISTS clause and a subquery with GROUP BY and HAVING clauses. The bottom of the window shows a toolbar with icons for table, chart, and file operations.

```
6 --3b
7 SELECT T1.id, T1.course_id
8 FROM takes T1
9 WHERE NOT EXISTS (
10     SELECT T1.id, T1.course_id
11     FROM takes T1
12     GROUP BY T1.id, T1.course_id
13     EXCEPT
14     SELECT T2.id, T2.course_id
15     FROM takes T2
16     GROUP BY T2.id, T2.course_id
17     HAVING COUNT(*) = 2
18 );
19
```

:C

```

11
12
13 SELECT DISTINCT(ss.id), ss.name, ss.dept_name, ss.tot_cred
14 FROM
15   (SELECT s.id, s.name, s.dept_name, s.tot_cred
16    FROM student s
17    WHERE s.dept_name = 'Comp. Sci.') AS ss
18 WHERE EXISTS
19   (SELECT *
20    FROM course c
21    WHERE c.dept_name = 'Comp. Sci.'
22    AND c.course_id NOT IN (SELECT t.course_id
23                           FROM takes t
24                           WHERE t.id = ss.id))
25

```

id	name	dept_name	tot_cred
66054	Crick	Comp. Sci.	86
19791	Vanrell	Comp. Sci.	61
25525	Moreira	Comp. Sci.	107
98830	Frolova	Comp. Sci.	13
88085	Bouamama	Comp. Sci.	8
36379	Triebel	Comp. Sci.	115
90663	Sram	Comp. Sci.	119
24010	Brookh	Comp. Sci.	65
31690	Bondi	Comp. Sci.	110
84845	Zuo	Comp. Sci.	81
65258	Tian	Comp. Sci.	3
16753	Kuwadak	Comp. Sci.	67
62549	Komatsu	Comp. Sci.	74
.....	...	...	...

:D

```
26
27
28 SELECT t.course_id, c.title, COUNT(s.id) AS num_students
29 FROM takes t JOIN course c ON t.course_id = c.course_id
30      JOIN student s ON t.id = s.id
31 WHERE t.semester = 'Fall' AND t.year = 2003
32 GROUP BY t.course_id
33 HAVING COUNT(s.id) > 300
34 ORDER BY num_students DESC;
35
```

course_id	title	num_students
974	Astronautics	321
748	Tort Law	318
400	Visual BASIC	312
603	Care and Feeding of Cats	306

```

35
36
37 SELECT s.*,c.title,t.grade ,CASE WHEN t.grade IN ('A ','B ','A+','A-','B-','B+') THEN 'PASS'
38        ELSE 'FAIL'
39        END AS pass_fail
40 FROM takes t JOIN student s ON t.ID=s.ID
41      JOIN course c ON t.course_id=c.course_id
42 WHERE t.semester='Spring' AND t.year = 2004
43

```

ID	name	dept_name	tot_cred	title	grade	pass_fail
73492	Hwang	Mech. Eng.	23	How to Groom your Cat	A	PASS
760	Liedm	Geology	100	Plastics	B	PASS
74016	Moei	Elec. Eng.	15	International Practicum	A-	PASS
52866	Loull	Math	30	Death and Taxes	B	PASS
66356	Xi	Elec. Eng.	44	Plastics	B+	PASS
4248	Wright	Finance	63	Plastics	C	FAIL
15883	Marques	Math	24	Plastics	C	FAIL
1110	Tzeng	Civil Eng.	23	International Practicum	B	PASS
65676	Aufr	Astronomy	93	Death and Taxes	A	PASS
47001	Correia	Comp. Sci.	63	Death and Taxes	C	FAIL
92703	Makinen	Cybernetics	29	Plastics	B-	PASS
45200	Kitagawa	Elec. Eng.	125	Death and Taxes	C+	FAIL
90779	Lenhart	Math	24	International Practicum	A-	PASS
94730	Neff	Geology	101	Plastics	C+	FAIL
33759	Mowbray	Psychology	44	How to Groom your Cat	B+	PASS
49503	Seaz	Finance	105	Death and Taxes	C-	FAIL
58172	Geißl	Astronomy	127	How to Groom your Cat	B-	PASS

f:



```

44 SELECT c.*
45 FROM section s JOIN course c ON s.course_id=c.course_id
46 GROUP BY s.course_id
47 HAVING COUNT(*)<3
48
49
-- =====

```

course_id	title	dept_name	credits
105	Image Processing	Astronomy	3
137	Manufacturing	Finance	3
158	Elastic Structures	Cybernetics	3
169	Marine Mammals	Elec. Eng.	3
192	Drama	Languages	4
200	The Music of the Ram...	Accounting	4
237	Surfing	Cybernetics	3
239	The Music of the Ram...	Physics	4
242	Rock and Roll	Marketing	3
258	Colloid and Surface C...	Math	3
270	Music of the 90s	Math	4
274	Corporate Law	Comp. Sci.	4
304	Music 2 New for your I...	Finance	4
313	International Trade	Marketing	3
319	World History	Finance	4
334	International Trade	Athletics	3
338	Graph Theory	Psychology	3
345	Race Car Driving	Accounting	4

```

50 WITH avg_units AS
51   (SELECT AVG(num) AS avg_courses
52    FROM
53     (SELECT s.id, s.name, COUNT(t.course_id) AS num
54      FROM student s JOIN takes t ON s.id = t.id
55      WHERE t.semester = 'Spring' AND t.year = 2005
56      GROUP BY s.id)
57   )
58 SELECT s.*, COUNT(t.course_id)
59 FROM student s JOIN takes t ON s.id = t.id
60      JOIN avg_units ON 1=1
61 WHERE t.semester = 'Spring' AND t.year = 2005 AND
62        s.dept_name = 'Comp. Sci.'
63 GROUP BY s.id, s.name
64 HAVING COUNT(t.course_id) > avg_units.avg_courses;
65
66

```

ID	name	dept_name	tot_cred	COUNT(t.course_id)
38696	Spadon	Comp. Sci.	118	2
43130	Yong	Comp. Sci.	123	2
44206	Gilmour	Comp. Sci.	75	2
46762	Bier	Comp. Sci.	71	2
63489	Enokib	Comp. Sci.	81	2
76246	Pettersen	Comp. Sci.	81	2
83136	Caporali	Comp. Sci.	61	3
87439	Scheffer	Comp. Sci.	93	2
9495	Crimm	Comp. Sci.	68	2

:h

```
66
67 SELECT c1.course_id, SUM(c3.credits) AS sum
68 FROM course c1 JOIN prereq c2 ON c1.course_id = c2.course_id JOIN course c3 ON c2.prereq_id=c3.course_id
69 GROUP BY c1.course_id
```

! course_id	sum
158	3
169	3
209	3
236	3
239	3
241	3
258	3
272	3
276	3
349	3
376	3
392	3
399	3
457	3
486	3
558	3
559	3
582	3
591	3

:A

```

74 --4a
75 UPDATE instructor
76 SET salary = salary*1.1
77 WHERE dept_name = 'Comp. Sci.';
78
79 SELECT *
80 FROM instructor
81 WHERE dept_name = 'Comp. Sci.'
82
83
84
85
86
87

```

ID	name	dept_name	salary
34175	Bondi	Comp. Sci.	127016.02100000001
3335	Bourrier	Comp. Sci.	88877.61300000001

B: لیست استادها قبل دستور ۵۰ تا

```

83 --b
84 SELECT COUNT(*)
85 FROM instructor
86
87 INSERT INTO instructor (id, name, dept_name, salary)
88 SELECT id, name, dept_name, 10000000
89 FROM student
90 WHERE tot_cred > 100;
91
92
93

```

```

COUNT(*)
50

```

لیست استادان بعد دستور ۵۱۳ تا

```

83 --b
84 SELECT COUNT(*)
85 FROM instructor
86
87 INSERT INTO instructor (id, name, dept_name, salary)
88 SELECT id, name, dept_name, 10000000
89 FROM student
90 WHERE tot_cred > 100;
91
92 SELECT COUNT(*)
93 FROM instructor
94
95 --c

```

COUNT(\*)  
 513

C: قبل آپدیت

```

97 SELECT i.name , s.building , s.room_number
98 FROM instructor i JOIN teaches t ON i.ID = t.ID JOIN section s ON t.course_id=s.course_id AND t.sec_id=s.sec_id AND t.semester =s.semester AND t.year =
99 WHERE i.name = 'Romero' AND tt.day IN ('S','M') AND tt.start_hr<12
100
101
102
103
104
105
106
107
108
109 --4d
110

```

name	building	room_number
Romero	Chandler	375
Romero	Taylor	183
Romero	Drown	757

بعد D:

```

97
98 SELECT t.course_id,t.sec_id,t.semester,t.year
99 FROM instructor i JOIN teaches t ON i.ID = t.ID JOIN section s ON t.course_id=s.course_id AND t.sec_id=s.sec_id AND t.semester =s.semester AND t.yea
100 WHERE i.name = 'Romero' AND tt.day IN ('S','M') AND tt.start_hr<12
101
102 UPDATE section
103 SET building = 'Kharazmi', room_number = '202'
104 WHERE (course_id , sec_id , semester , year) IN (
105 SELECT t.course_id,t.sec_id,t.semester,t.year
106 FROM instructor i JOIN teaches t ON i.ID = t.ID JOIN section s ON t.course_id=s.course_id AND t.sec_id=s.sec_id AND t.semester =s.semester AND t.yea
107 WHERE i.name = 'Romero' AND tt.day IN ('S','M') AND tt.start_hr<12)
108
109 SELECT i.* , s.*
110 FROM instructor i JOIN teaches t ON i.ID = t.ID JOIN section s ON t.course_id=s.course_id AND t.sec_id=s.sec_id AND t.semester =s.semester AND t.yea
111 WHERE i.name = 'Romero' AND tt.day IN ('S','M') AND tt.start_hr<12
112
113
114
115
116
---
```

i	ID	name	dept_name	salary	course_id	sec_id	semester	year	building	room_num...	time_slot_id
	43779	Romero	Astronomy	79070.08	105	1	Fall	2009	Kharazmi	202	C
	43779	Romero	Astronomy	79070.08	105	2	Fall	2002	Kharazmi	202	C
	43779	Romero	Astronomy	79070.08	476	1	Fall	2010	Kharazmi	202	C

:d

```

111 WHERE i.name = 'Romero' AND tt.day IN ('S','M') AND tt.start_hr<12
112 --4d
113 SELECT c.* , s.*
114 FROM course c JOIN section s ON s.course_id = c.course_id
115
116 DELETE FROM course
117 WHERE course_id NOT IN
118 (SELECT DISTINCT course_id
119 FROM section
120 WHERE year >= 2019);
121
122
```

i	course_id	title	dept_name	credits	course_id	sec_id	semester	year	building	room_num...	time_slot_id
	313	International ...	Marketing	3	313	1	Fall	2010	Chandler	804	N
	747	International ...	Comp. Sci.	4	747	1	Spring	2004	Gates	314	K
	443	Journalism	Physics	4	443	1	Spring	2010	Whitman	434	O
	893	Systems Soft...	Cybernetics	3	893	1	Fall	2007	Fairchild	145	B
	663	Geology	Psychology	3	663	1	Spring	2005	Fairchild	145	D
	457	Systems Soft...	History	3	457	1	Spring	2001	Saucon	844	D
	445	Biostatistics	Finance	3	445	1	Spring	2001	Alumni	547	J

بعد اجرا

```

116 DELETE FROM course
117 WHERE course_id NOT IN
118     (SELECT DISTINCT course_id
119      FROM section
120      WHERE year >= 2019);
121
122 SELECT c.*, s.*
123 FROM course c JOIN section s ON s.course_id = c.course_id
124
125
126
127

```

سوال ۵:

A:

SQLite

```

130
131 CREATE VIEW v_students_avg_grade AS
132 SELECT s.ID, s.name, a.i_ID, i.name,
133        AVG(CASE WHEN t.grade = 'A+' THEN 100
134              WHEN t.grade = 'A' THEN 95
135              WHEN t.grade = 'A-' THEN 90
136              WHEN t.grade = 'B+' THEN 85
137              WHEN t.grade = 'B' THEN 80
138              WHEN t.grade = 'B-' THEN 75
139              WHEN t.grade = 'C+' THEN 85
140              WHEN t.grade = 'C' THEN 80
141              WHEN t.grade = 'C-' THEN 75
142              ELSE 0 END) AS avg_grade
143 FROM student s LEFT JOIN advisor a ON s.ID = a.s_ID
144               LEFT JOIN takes t ON s.ID = t.ID
145               LEFT JOIN instructor i ON i.ID = a.i_ID
146 GROUP BY s.ID, s.name, a.i_ID, i.name;
147
148
149 SELECT * FROM v_students_avg_grade
150

```

ID	name	I_ID	name:1	avg_grade
1000	Manber	16807	Yazdi	64.23076923076923
10033	Zelty	64871	Gutierrez	44.77272727272727
10076	Duan	6569	Mingoz	70.35714285714286
1018	Colin	79653	Levine	47.04545454545455
10204	Mediratta	65931	Pimenta	48.333333333333336
10267	Rzecz	59795	Desyl	62.72727272727273
10269	Hilberg	15347	Bawa	67.5
10454	Ugarte	37687	Arias	51.944444444444444
10481	Grosch	19368	Wieland	49.11764705882353
10577	Klarn	36007	Klarin	65

B:راه اول

SQLite

```

151 --5b
152 CREATE VIEW v_top_students_by_dept AS
153 WITH top_students AS (
154     SELECT s.dept_name, s.ID, s.name,
155            AVG(CASE WHEN t.grade = 'A+' THEN 100
156                  WHEN t.grade = 'A' THEN 95
157                  WHEN t.grade = 'A-' THEN 90
158                  WHEN t.grade = 'B+' THEN 85
159                  WHEN t.grade = 'B' THEN 80
160                  WHEN t.grade = 'B-' THEN 75
161                  WHEN t.grade = 'C+' THEN 85
162                  WHEN t.grade = 'C' THEN 80
163                  WHEN t.grade = 'C-' THEN 75
164                  ELSE 0 END) AS gpa
165     FROM student s JOIN takes t ON s.ID = t.ID
166     GROUP BY s.ID, s.dept_name
167     HAVING SUM(CASE WHEN t.grade IN('A%', 'B%', 'C%') THEN 1 ELSE 0 END) = 0
168     ORDER BY s.dept_name, gpa DESC
169 )
170 SELECT *
171 FROM (SELECT dept_name, ID, name, gpa,
172            ROW_NUMBER() OVER (PARTITION BY dept_name
173                               ORDER BY gpa DESC) AS rnum
174     FROM top_students) x
175 WHERE rnum <= 3;
176 SELECT *
177 FROM v_top_students_by_dept
178

```

dept_name	ID	name	gpa	rnum
Athletics	15487	Januszewski	72.25	2
Athletics	3640	Karniel	72.08333333333333	3
Biology	80990	Strzem	74.64285714285714	1
Biology	8843	Papakir	73.92857142857143	2
Biology	67407	Yoneda	73.52941176470588	3
Civil Eng.	17128	Chuon	79	1



SQLite

```

178 ---way2
179 CREATE VIEW top_students_dense_rank AS
180 SELECT dept_name, admission_year, name, gpa, dense_rank
181 FROM
182 (SELECT dept_name, tot_cred AS admission_year, name, AVG(grade) AS gpa,
183      DENSE_RANK() OVER (PARTITION BY dept_name, tot_cred
184      ORDER BY AVG(CASE WHEN grade = 'A+' THEN 100
185      WHEN grade = 'A' THEN 95
186      WHEN grade = 'A-' THEN 90
187      WHEN grade = 'B+' THEN 85
188      WHEN grade = 'B' THEN 80
189      WHEN grade = 'B-' THEN 75
190      WHEN grade = 'C+' THEN 85
191      WHEN grade = 'C' THEN 80
192      WHEN grade = 'C-' THEN 75
193      ELSE 0 END) DESC) AS dense_rank
194 FROM student JOIN takes ON student.id = takes.id
195 GROUP BY dept_name, tot_cred, name) derived_table
196 WHERE dense_rank <= 3;
197
198
199 SELECT *
200 FROM v_top_students_by_dept
201
202 SELECT *
203 FROM top_students_dense_rank
204

```

dept_name	ID	name	gpa	rnum
Accounting	83592	Benkov	87.5	1
Accounting	30222	Lepp	79.25	2
Accounting	35523	Yamamoto	78.33333333333333	3
Astronomy	57941	Kleinberg	82.5	1
Astronomy	57107	Janssen	80.35714285714286	2
Astronomy	31101	Lhomme	76.92307692307692	3
Athletics	94371	Milner	76.36363636363636	1

C:

SQLite

```
176 SELECT *
177 FROM v_top_students_by_dept
178 --c
179 CREATE VIEW v_dept_courses AS
180 SELECT d.dept_name, sec.year, sec.semester,
181        SUM(c.credits) AS total_units
182 FROM department d JOIN course c ON d.dept_name = c.dept_name
183        JOIN section sec ON c.course_id = sec.course_id
184 GROUP BY d.dept_name, sec.year, sec.semester
185 ORDER BY d.dept_name, sec.year, sec.semester;
186
187 SELECT * FROM v_dept_courses
188
189
190
191
```

dept_name	year	semester	total_units
Accounting	2002	Fall	4
Accounting	2003	Fall	3
Accounting	2003	Spring	3
Accounting	2004	Spring	3
Accounting	2007	Spring	7
Accounting	2008	Spring	4
Astronomy	2002	Fall	3
Astronomy	2007	Fall	4
Astronomy	2009	Fall	3
Astronomy	2010	Fall	4
Athletics	2003	Fall	4
Athletics	2004	Fall	4

SQLite

```

190 CREATE VIEW v_student_prof_avg AS
191 SELECT s.name AS student, i.name AS professor,
192        AVG(CASE WHEN t.grade = 'A+' THEN 100
193              WHEN t.grade = 'A' THEN 95
194              WHEN t.grade = 'A-' THEN 90
195              WHEN t.grade = 'B+' THEN 85
196              WHEN t.grade = 'B' THEN 80
197              WHEN t.grade = 'B-' THEN 75
198              WHEN t.grade = 'C+' THEN 85
199              WHEN t.grade = 'C' THEN 80
200              WHEN t.grade = 'C-' THEN 75
201              ELSE 0 END) AS avg_grade
202 FROM student s JOIN takes t ON s.ID = t.ID
203              JOIN teaches te ON t.course_id = te.course_id AND t.sec_id = te.sec_id AND
204                          t.semester = te.semester AND t.year = te.year
205              JOIN instructor i ON i.ID = te.ID
206 GROUP BY s.name, i.name;
207
208 SELECT *
209 FROM v_student_prof_avg
210

```

i	student	professor	avg_grade
	Aarde	Bawa	0
	Aarde	Bietzk	90
	Aarde	Bondl	37.5
	Aarde	Bourrier	91.66666666666667
	Aarde	D'Agostino	0
	Aarde	Dale	75
	Aarde	Gustafsson	90
	Aarde	Kean	42.5
	Aarde	Mahmoud	42.5
	Aarde	Minnert	37.5