



دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

: عنوان

پروژه درس طراحی پایگاه داده‌ها

Database-Design Course Project

نگارش

ملیکا علیزاده - ثمین اکبری - معین آعلی

استاد

مهردی آخی

نیمسال دوم ۰۲-۰۳

تقسیم‌بندی پروژهتسک‌های مربوط به عضو اول: [ملیکا علیزاده](#)

۱. ساختن جداول در در دیتابیس

۲. نرم‌ال سازی

۳. تولید داده فیک

۴. اصلاح کوئری‌ها

۵. ساختن index

تسک‌های مربوط به عضو دوم: [شمین اکبری](#)

۱. تغییر دادن روابط ERD

۲. نرم‌ال سازی

۳. ساختن index

۴. امتیازی بخش Mongo

۵. اصلاح کوئری‌ها

تسک‌های مربوط به عضو سوم: [معین آعلی](#)

۱. نوشتمن مستندات در قالب LaTeX

۲. بخش امتیازی مربوط به API

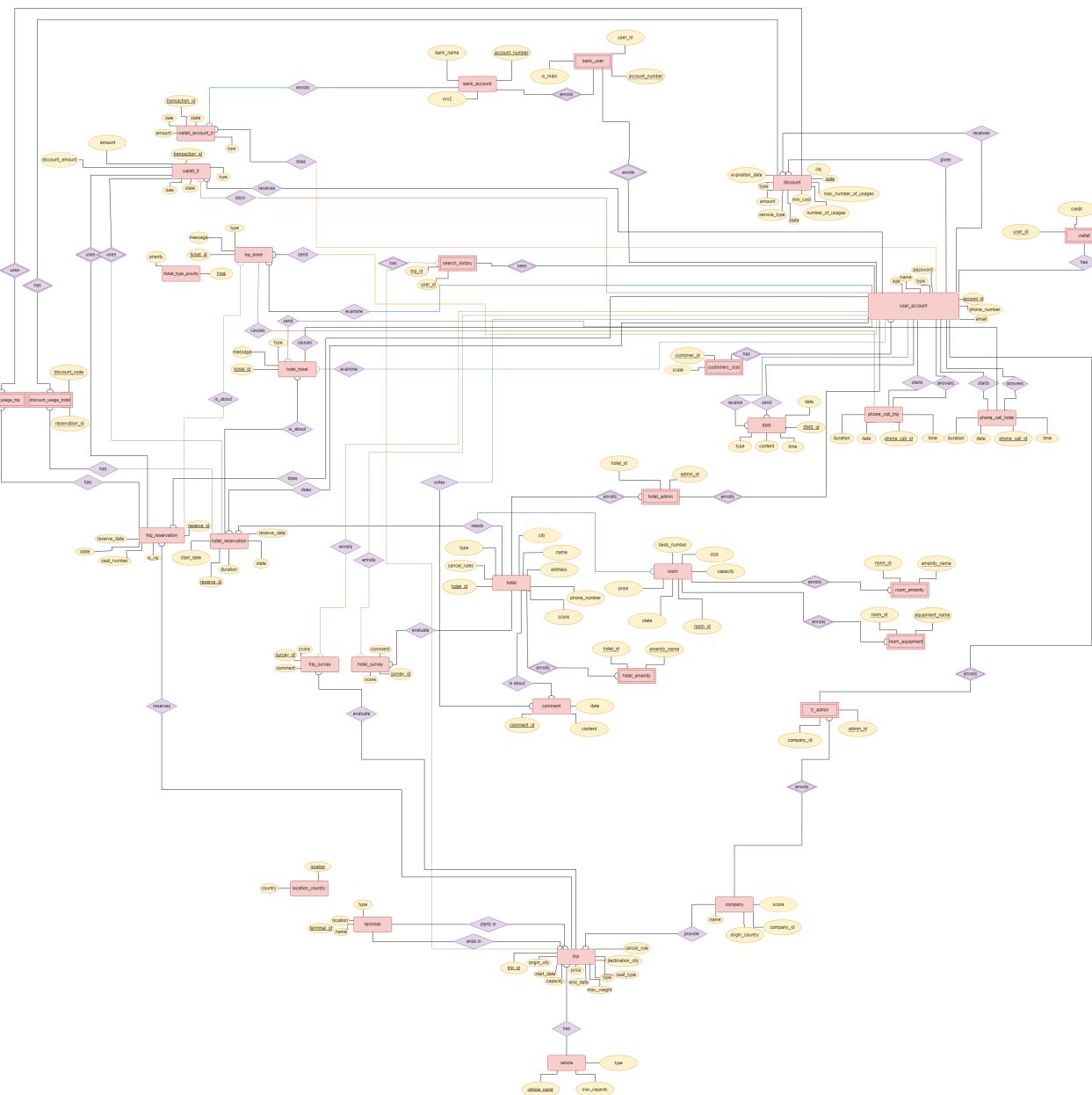
۳. نوشتمن کوئری‌ها

۴. تولید داده فیک

جزئیات تقسیم‌بندی کارها داخل این [Github Issue](#) موجود است.

نمودار رابطه - موجودیت

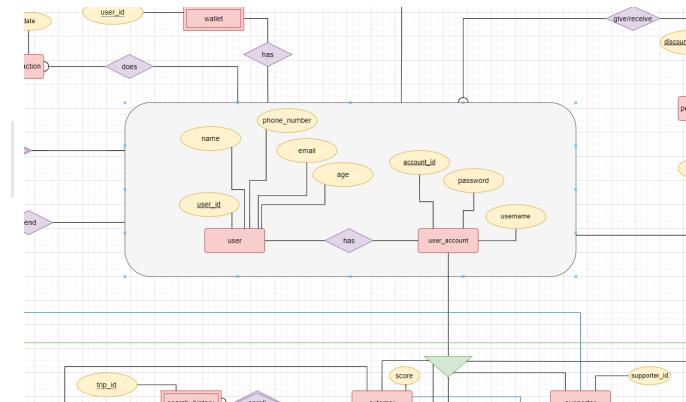
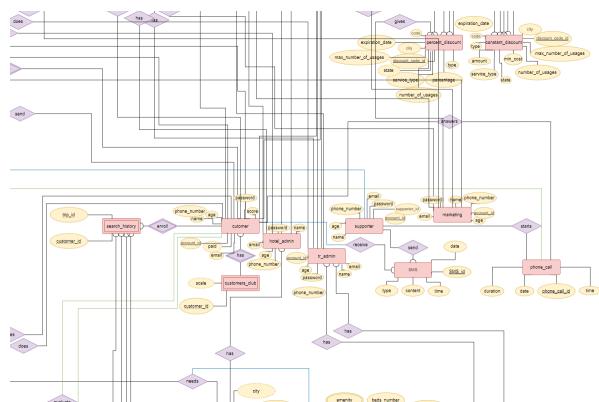
نمودار مربوط به فاز دوم پس از اعمال تغییرات مورد نیاز بر روی نمودار فاز ۱ به این صورت است:



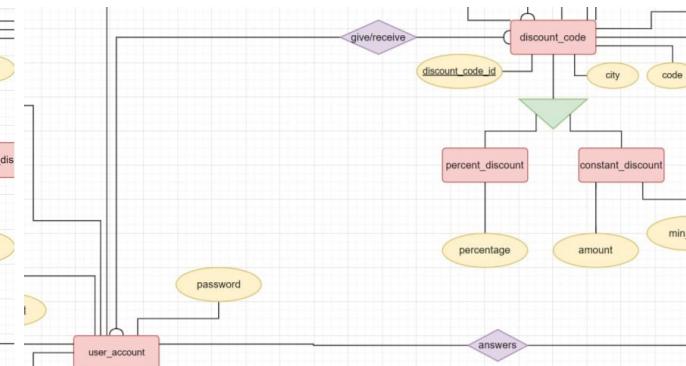
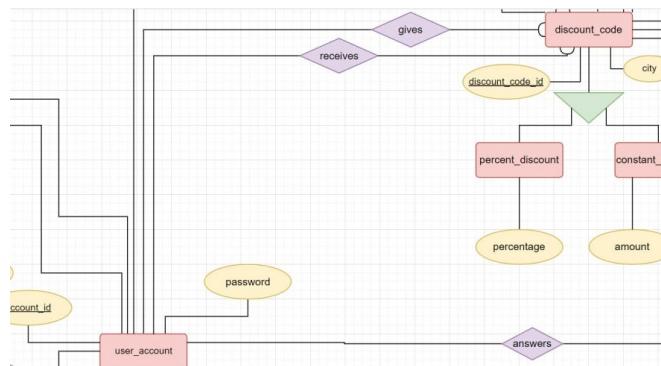
فایل Draw.io مربوط به ERD فاز دوم پروژه، داخل پیوست قرار دارد.

تغییرات نمودار در راستای انطباق بر SQL و نرم‌مال‌تر سازی

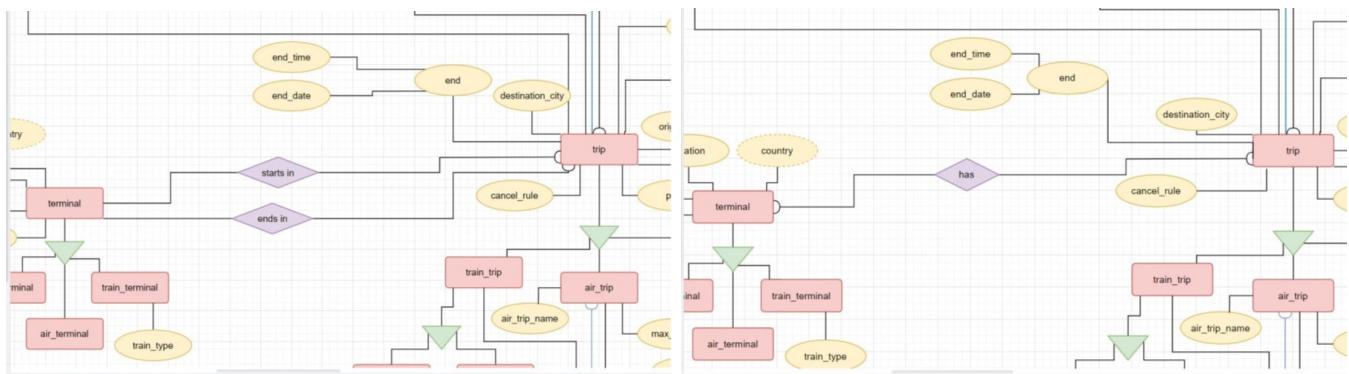
در تمامی عکس‌های این بخش، عکس سمت راست معادل پیش از تغییرات است و عکس سمت چپ معادل پس از تغییرات است. دلیل برخی از تغییرات در ادامه به دلیل تکراری بودن ذکر نشده است.



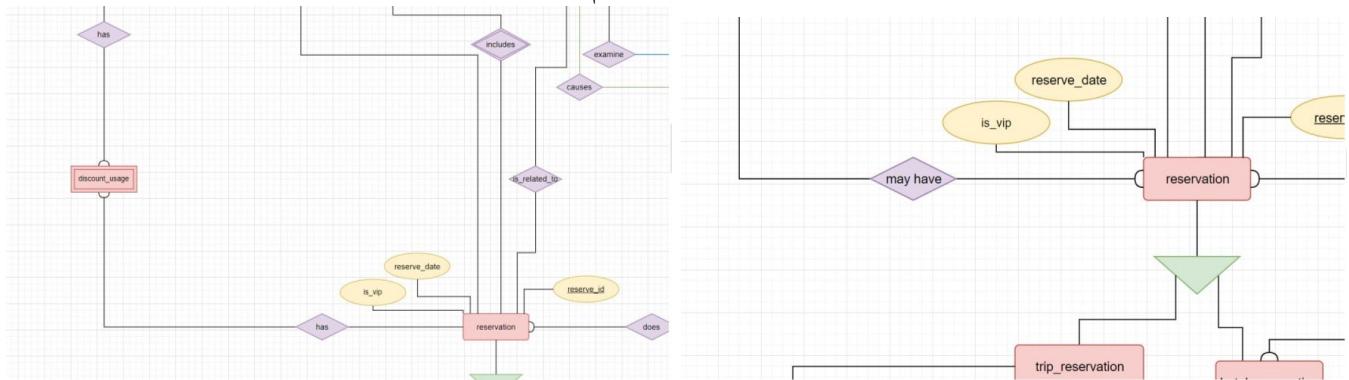
اگریگیشن در SQL وجود ندارد. بنابراین باید شکسته شود. در اینجا تصمیم بر این شد که کاربر و حساب کاربری به دلیل داشتن رابطه ۱:۱ و ویژگی‌های تقریباً مرتبط تبدیل به یک موجودیت شوند و تمام روابطی که به این اگریگیشن متصل است، به userAccount وصل شود.



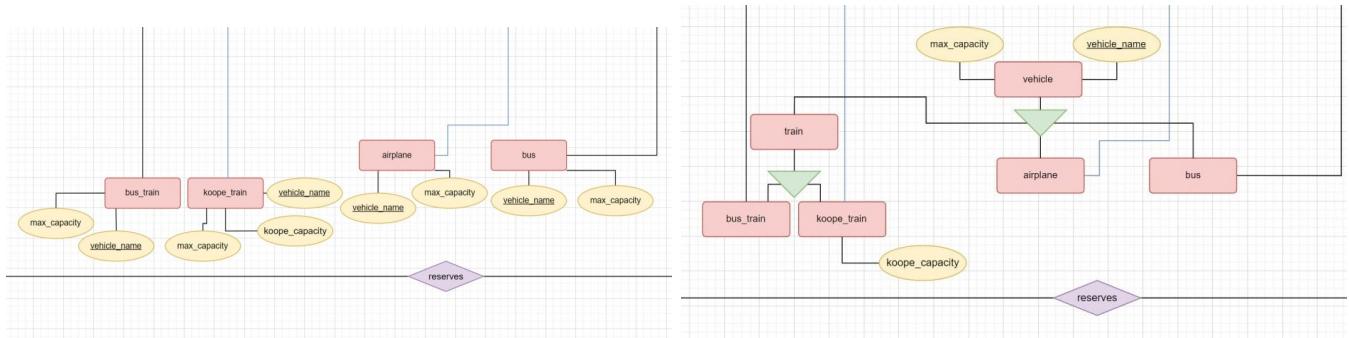
در گام بعد باید همه روابط manyToMany شکسته شوند. در اینجا رابطه بین userAccount و discountCode باید شکسته شود. برای این کار این رابطه را به دو رابطه تبدیل می‌کنیم که یکی از آنها مربوط به دریافت‌کننده و دیگری مربوط به اهداکننده کد تخفیف است.



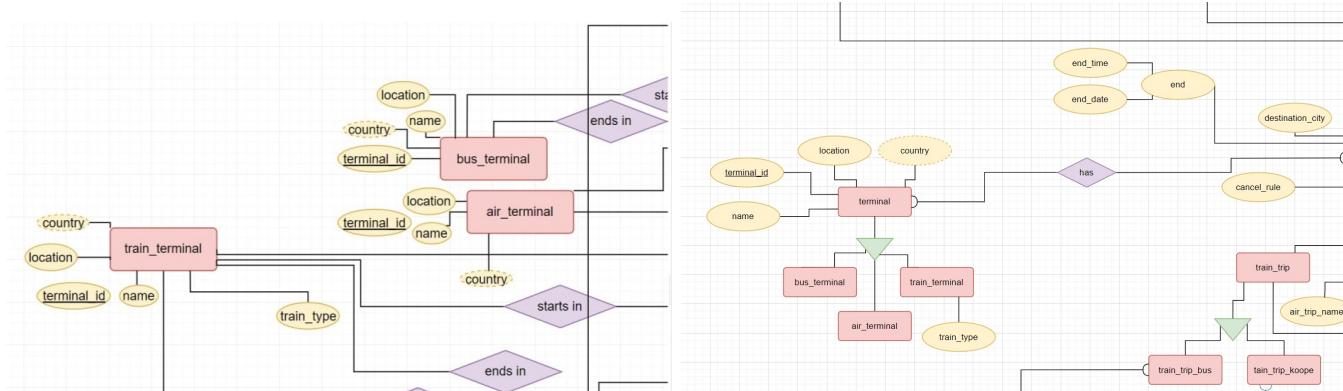
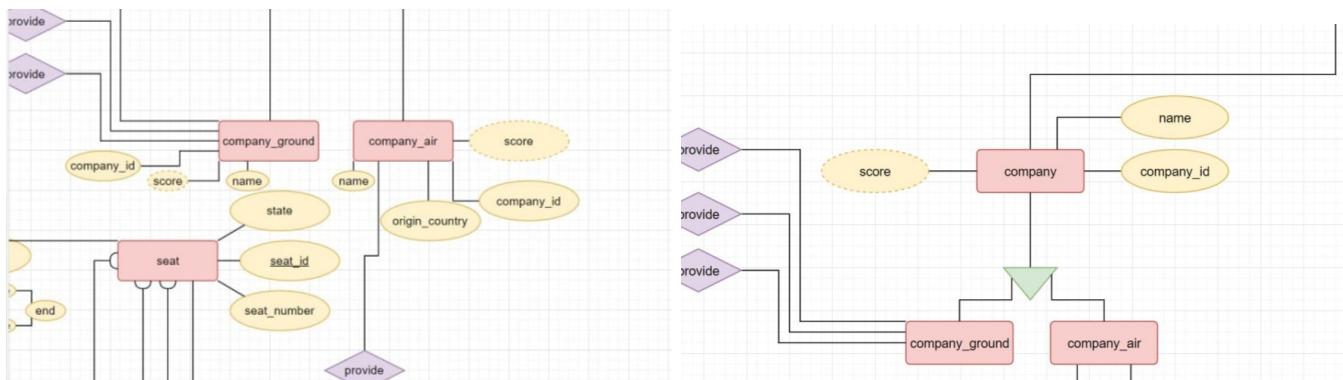
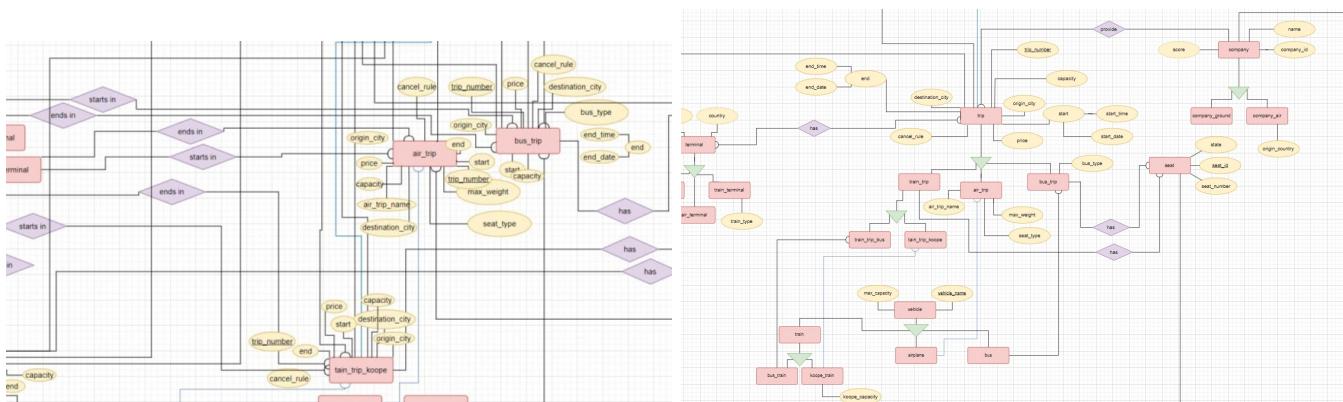
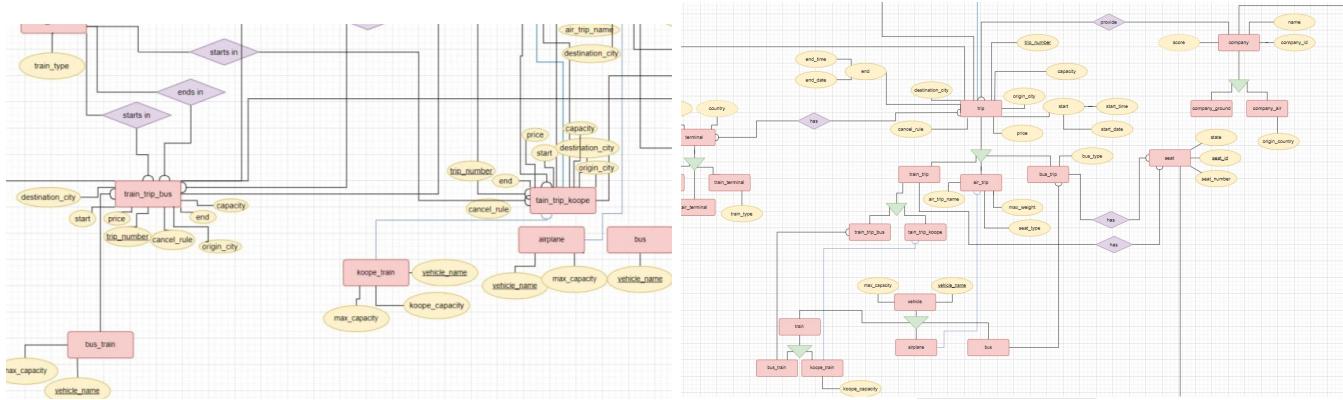
در اینجا بین **terminal** و **trip** رابطه m:n برقرار بود زیرا از هر ترمینال چند سفر انجام می‌شود و هر سفر یک ترمینال مبدأ و یک ترمینال مقصد دارد. حالا باید این رابطه را تبدیل به دو رابطه اکنیم که یکی مبدأ را مشخص می‌کند و یکی مقصد را.

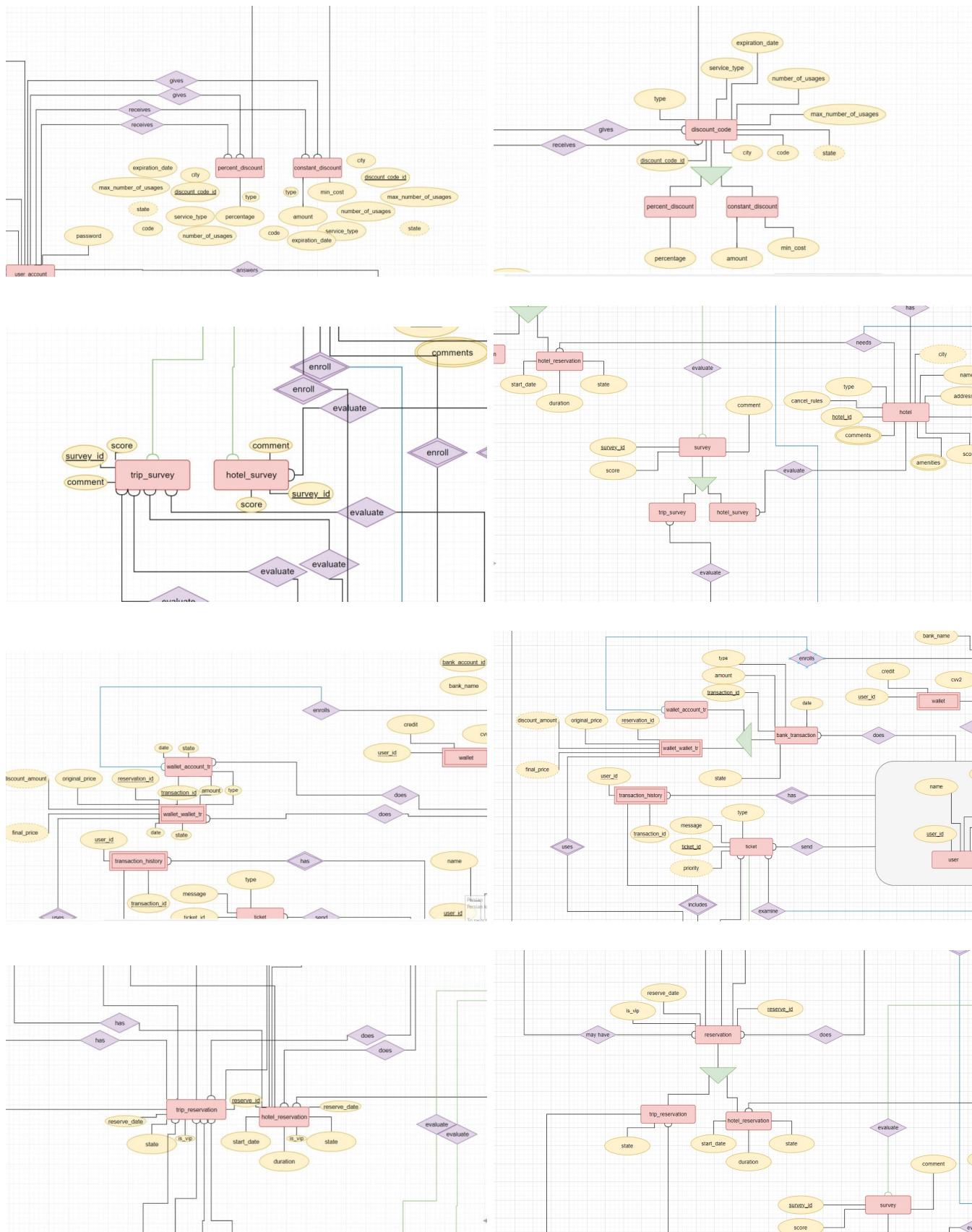


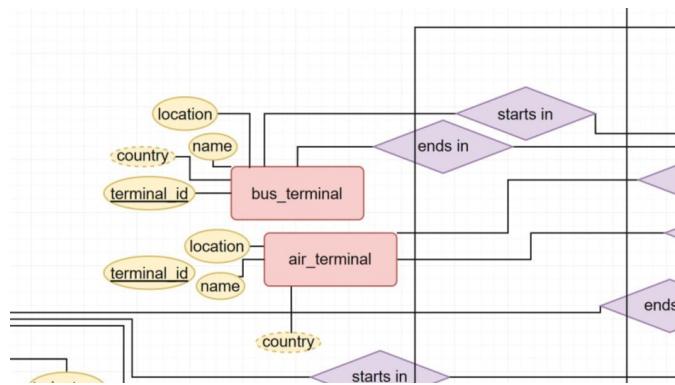
رابطه بین **discountCode** و **reservation** هم m:n بود چون هر کد تخفیف ممکن است بر چند رزرو اعمال شود و هر رزرو چند کد تخفیف داشته باشد. برای حل مشکل می‌توانیم یک موجودیت ضعیف **discountUsage** داشته باشیم که کلید خارجی به این دو موجودیت داشته باشد و مشخص کند هر استفاده‌ای از کد تخفیف مربوط به چه کد تخفیفی و برای چه رزروی است.



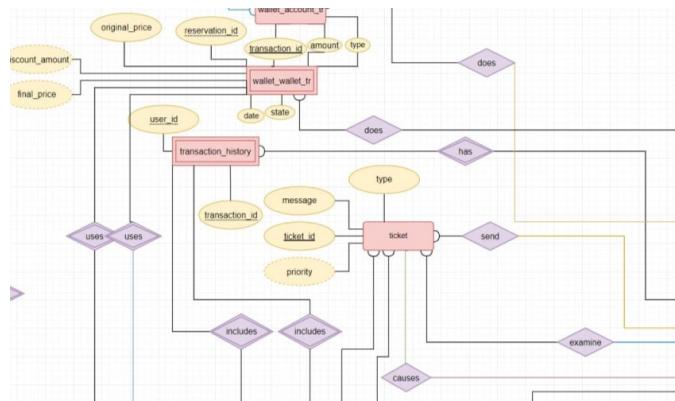
مورد دیگری که در SQL وجود ندارد ولی در نمودار ما بود، specialization است. برای حل این مشکل موجودیت کلی‌تر را حذف می‌کنیم، و تمام **attribute** و روابط آن را به همه موجودیت‌های جزئی آن نسبت می‌دهیم.



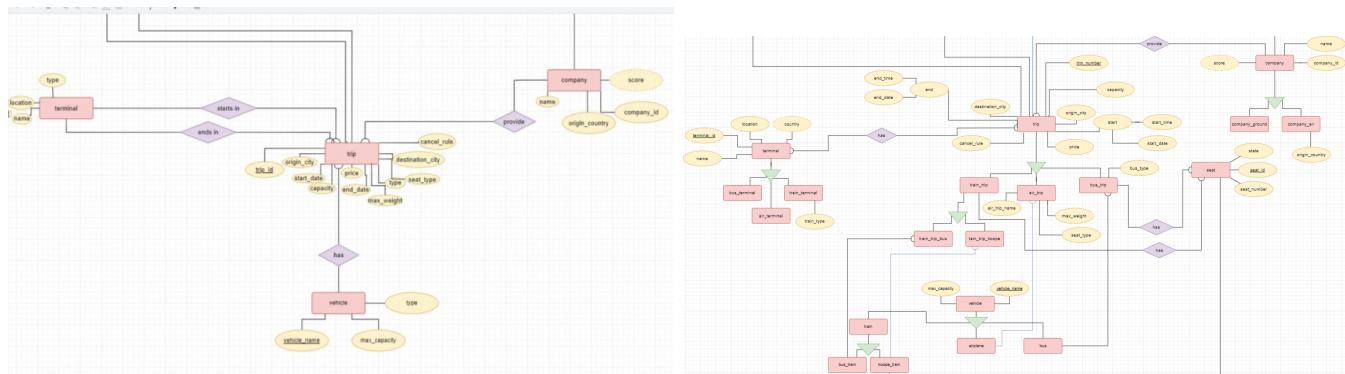




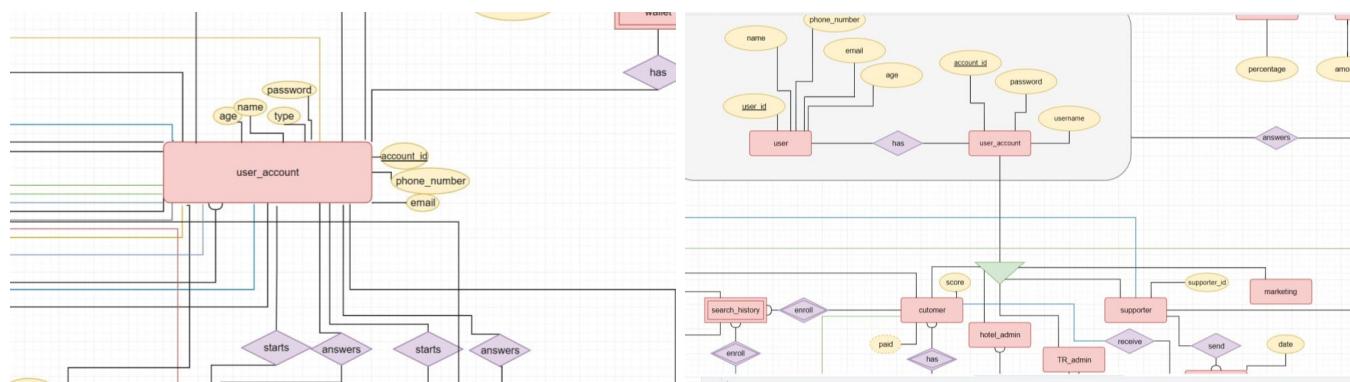
در همه انواع ترمینال لوکیشن میتواند کشور را مشخص کند، و لوکیشن یک ۳nf باید این مورد رفع شود. یک موجودیت ضعیف برای این مورد میسازیم و ویژگی کشور را از ترمینال حذف میکنیم.



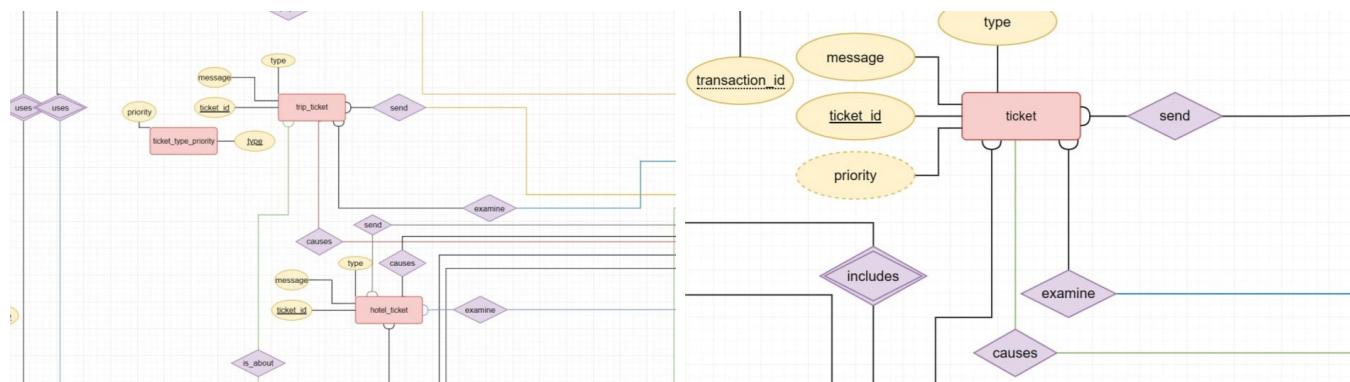
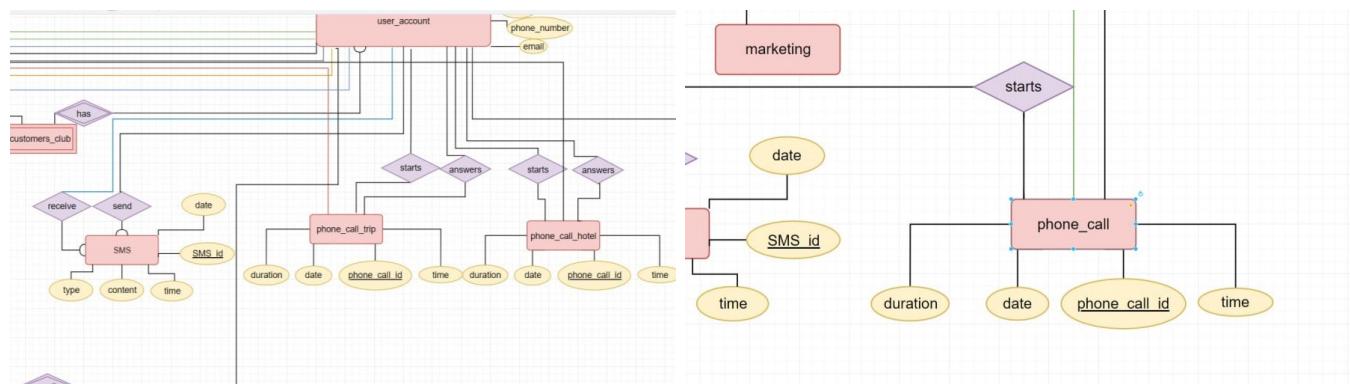
در اینجا هم بین type و priority وجود دارد. به همین دلیل این دو را در جدولی جدا درج کردیم که اصول ۳nf برقرار باشد.



در حین پیاده‌سازی جداول متوجه شدیم که به صورت انواع مختلفی از یک موجودیت جدا کردیم (specialization)، میتوانند همگی یک موجودیت باشند که نوعشان در اtributیوتی به نام type ذخیره شود:



همچنین دریافتیم که نیاز است برخی از موجودیت‌ها به دو موجودیت جداگانه شکسته شوند. دلیل این موضوع، **foreign key** بود که مشخص نبود باید دقیقاً به کدام موجودیت اشاره کنند (دو حالت داشتند).



ساخت جداول در Postgres

جدائل دیتابیس ما از روی ERD اصلاح شده، مطابق دستورات زیر ساخته شدند:
فایل sql مربوط به create-table.sql داخل پوست قرار دارد.

```

1 CREATE TABLE user_account
2 (
3     account_id    SERIAL PRIMARY KEY,
4     type          TEXT,
5     name          TEXT,
6     password      TEXT,
7     age           INT,
8     phone_number TEXT,
9     email         TEXT
10 );
11
12 CREATE TABLE customer_club
13 (
14     customer_id  INT not null PRIMARY KEY,
15     scale         INT,
16     CONSTRAINT fk_customer
17         FOREIGN KEY (customer_id)
18             REFERENCES user_account (account_id)
19 );
20
21 CREATE TABLE SMS
22 (
23     SMS_id        SERIAL not null PRIMARY KEY,
24     sender_id     INT    not null,
25     receiver_id   INT    not null,
26     date          DATE,
27     type          TEXT,
28     content       TEXT,
29     time          TIME,
30     CONSTRAINT fk_sender
31         FOREIGN KEY (sender_id)
32             REFERENCES user_account (account_id),
33     CONSTRAINT fk_receiver
34         FOREIGN KEY (receiver_id)
35             REFERENCES user_account (account_id)
36 );
37
38
39 CREATE TABLE wallet
40 (
41     user_id       INT not null PRIMARY KEY,
42     credit        DECIMAL,
43     CONSTRAINT fk_user
44         FOREIGN KEY (user_id)
45             REFERENCES user_account (account_id)
46 );
47
48 CREATE TABLE discount
49 (
50     code          VARCHAR(255) not null PRIMARY KEY,
51     user_id       INT          not null,
52     city          TEXT,
53     expiration_date DATE,
54     max_number_of_usages INT,
55     service_type  TEXT,
56     amount        DECIMAL,
57     type          TEXT,
58     min_cost      DECIMAL,
59     number_of_usage INT,
60     CONSTRAINT fk_user
61         FOREIGN KEY (user_id)
62             REFERENCES user_account (account_id)
63 );
64
65
66

```

```

96 CREATE TABLE bank_account
97 (
98     account_number INT not null PRIMARY KEY,
99     bank_name      TEXT,
100    CVV2          INT
101 );
102
103 CREATE TABLE bank_user
104 (
105     account_number INT not null PRIMARY KEY,
106     user_id       INT,
107     is_main        boolean default false,
108     CONSTRAINT fk_account_number
109         FOREIGN KEY (account_number)
110             REFERENCES bank_account (account_number),
111     CONSTRAINT fk_user
112         FOREIGN KEY (user_id)
113             REFERENCES user_account (account_id)
114 );
115
116 CREATE TABLE hotel
117 (
118     hotel_id      SERIAL PRIMARY KEY,
119     type          INT          not null,
120     name          TEXT         not null,
121     address       TEXT         not null,
122     city          TEXT         not null,
123     cancel_rule   TEXT         not null,
124     score         float default 0 not null,
125     phone_number  TEXT         not null
126 );
127
128 CREATE TABLE hotel_admin
129 (
130     hotel_id INT not null,
131     admin_id INT not null,
132     PRIMARY KEY (hotel_id, admin_id),
133     CONSTRAINT fk_hotel
134         FOREIGN KEY (hotel_id)
135             REFERENCES hotel (hotel_id),
136     CONSTRAINT fk_admin
137         FOREIGN KEY (admin_id)
138             REFERENCES user_account (account_id)
139 );
140
141 CREATE TABLE hotel_amenity
142 (
143     hotel_id      INT          not null,
144     amenity_name varchar(255) not null,
145     PRIMARY KEY (hotel_id, amenity_name),
146     CONSTRAINT fk_hotel
147         FOREIGN KEY (hotel_id)
148             REFERENCES hotel (hotel_id)
149 );
150
151 CREATE TABLE room
152 (
153     room_id      SERIAL PRIMARY KEY,
154     hotel_id     INT NOT NULL,
155     state        boolean default False,
156     capacity     INT not null,
157     size          INT not null,
158     beds_number  INT not null,
159     price         DECIMAL,
160     CONSTRAINT fk_hotel
161         FOREIGN KEY (hotel_id)
162             REFERENCES hotel (hotel_id)
163 );
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1696
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1796
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1896
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2189
2190
2191
2192
2193
2194
2195
2196
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2279
2280
2281
228
```

```

137 CREATE TABLE room_amenity
138 (
139     room_id      INT          not null,
140     amenity_name varchar(255) not null,
141     PRIMARY KEY (room_id, amenity_name),
142     CONSTRAINT fk_room
143         FOREIGN KEY (room_id)
144             REFERENCES room (room_id)
145 );
146
147 CREATE TABLE room_equipment
148 (
149     room_id      INT          not null,
150     equipment_name varchar(255) not null,
151     PRIMARY KEY (room_id, equipment_name),
152     CONSTRAINT fk_room
153         FOREIGN KEY (room_id)
154             REFERENCES room (room_id)
155 );
156
157 CREATE TABLE company
158 (
159     company_id    SERIAL        not null PRIMARY KEY,
160     name          TEXT          not null,
161     score         float default 0 not null,
162     origin_country TEXT         not null
163 );
164
165 CREATE TABLE tr_admin
166 (
167     company_id INT not null,
168     admin_id   INT not null,
169     PRIMARY KEY (admin_id),
170     CONSTRAINT fk_admin
171         FOREIGN KEY (company_id)
172             REFERENCES company (company_id),
173     CONSTRAINT fk_admin
174         FOREIGN KEY (admin_id)
175             REFERENCES user_account (account_id)
176 );
177
178 CREATE TABLE comment
179 (
180     comment_id   SERIAL not null PRIMARY KEY,
181     hotel_id    INT    not null,
182     writer_id   INT    not null,
183     date        DATE,
184     content     TEXT,
185     CONSTRAINT fk_hotel
186         FOREIGN KEY (hotel_id)
187             REFERENCES hotel (hotel_id),
188     CONSTRAINT fk_writer
189         FOREIGN KEY (writer_id)
190             REFERENCES user_account (account_id)
191 );
192
193
194 CREATE TABLE terminal
195 (
196     terminal_id  SERIAL not null PRIMARY KEY,
197     name         TEXT,
198     location     TEXT,
199     type         TEXT
200 );
201
202 CREATE TABLE vehicle
203 (
204     vehicle_name TEXT not null PRIMARY KEY,
205     max_capacity INT,
206     type         VARCHAR(255)
207 );

```

```

CREATE TABLE trip
(
    trip_id          VARCHAR(255) not null PRIMARY KEY,
    origin_terminal_id INT        not null,
    destination_terminal_id INT        not null,
    vehicle_name     TEXT       not null,
    origin_city      TEXT       not null,
    price            DECIMAL,
    capacity          INT,
    destination_city TEXT,
    end_date          DATE,
    start_date        DATE,
    cancel_rule      TEXT,
    max_weight        DECIMAL,
    seat_type         TEXT,
    CONSTRAINT fk_vehicle
        FOREIGN KEY (vehicle_name)
            REFERENCES vehicle (vehicle_name),
    CONSTRAINT fk_origin_terminal
        FOREIGN KEY (origin_terminal_id)
            REFERENCES terminal (terminal_id),
    CONSTRAINT fk_destination_terminal
        FOREIGN KEY (destination_terminal_id)
            REFERENCES terminal (terminal_id)
);
;

CREATE TABLE location_country
(
    location TEXT not null PRIMARY KEY,
    country  TEXT not null
);
;

CREATE TABLE wallet_account_tr
(
    transaction_id    SERIAL not null PRIMARY KEY,
    bank_account_number INT,
    user_id           INT,
    date              DATE,
    token             TEXT,
    amount            DECIMAL,
    type              TEXT,
    state             TEXT,
    CONSTRAINT fk_account_number
        FOREIGN KEY (bank_account_number)
            REFERENCES bank_account (account_number),
    CONSTRAINT fk_user
        FOREIGN KEY (user_id)
            REFERENCES user_account (account_id)
);
;

CREATE TABLE wallet_tr
(
    transaction_id    SERIAL not null PRIMARY KEY,
    giver_id          INT,
    receiver_id        INT,
    original_price    DECIMAL,
    discount_amount   DECIMAL,
    date              DATE,
    amount            DECIMAL,
    type              TEXT,
    state             TEXT,
    CONSTRAINT fk_giver
        FOREIGN KEY (giver_id)
            REFERENCES user_account (account_id),
    CONSTRAINT fk_receiver
        FOREIGN KEY (receiver_id)
            REFERENCES user_account (account_id)
);
;

```

```

779 CREATE TABLE trip_reservation
780 (
781     reserve_id      SERIAL not null PRIMARY KEY,
782     transaction_id INT,
783     trip_number    VARCHAR(255),
784     user_id        INT,
785     reserve_date   DATE,
786     is_vip         boolean default false,
787     state          TEXT,
788     seat_number    INT,
789     CONSTRAINT fk_trip
790         FOREIGN KEY (trip_number)
791             REFERENCES trip (trip_id),
792     CONSTRAINT fk_admin
793         FOREIGN KEY (user_id)
794             REFERENCES user_account (account_id),
795     CONSTRAINT fk_transaction
796         FOREIGN KEY (transaction_id)
797             REFERENCES wallet_tr (transaction_id)
798 );
799
800 CREATE TABLE hotel_reservation
801 (
802     reserve_id      SERIAL not null PRIMARY KEY,
803     transaction_id INT,
804     hotel_id       INT,
805     room_id        INT,
806     user_id        INT,
807     reserve_date   DATE,
808     duration       INTERVAL,
809     state          TEXT,
810     start_date    DATE,
811     CONSTRAINT fk_hotel
812         FOREIGN KEY (hotel_id)
813             REFERENCES hotel (hotel_id),
814     CONSTRAINT fk_room
815         FOREIGN KEY (room_id)
816             REFERENCES room (room_id),
817     CONSTRAINT fk_admin
818         FOREIGN KEY (user_id)
819             REFERENCES user_account (account_id),
820     CONSTRAINT fk_transaction
821         FOREIGN KEY (transaction_id)
822             REFERENCES wallet_tr (transaction_id)
823 );
824
825 CREATE TABLE discount_usage_trip
826 (
827     discount_code  VARCHAR(255) not null,
828     reservation_id INT          not null,
829     PRIMARY KEY (discount_code, reservation_id),
830     CONSTRAINT fk_discount
831         FOREIGN KEY (discount_code)
832             REFERENCES discount (code),
833     CONSTRAINT fk_reservation
834         FOREIGN KEY (reservation_id)
835             REFERENCES trip_reservation (reserve_id)
836 );
837
838 CREATE TABLE discount_usage_hotel
839 (
840     discount_code  VARCHAR(255) not null,
841     reservation_id INT          not null,
842     PRIMARY KEY (discount_code, reservation_id),
843     CONSTRAINT fk_discount
844         FOREIGN KEY (discount_code)
845             REFERENCES discount (code),
846     CONSTRAINT fk_reservation
847         FOREIGN KEY (reservation_id)
848             REFERENCES hotel_reservation (reserve_id)
849 );

```

```

۷۰+
۷۰۱ CREATE TABLE trip_survey
(
۷۰۲     survey_id    SERIAL not null PRIMARY KEY,
۷۰۳     trip_number  VARCHAR(255),
۷۰۴     user_id      INT,
۷۰۵     comment      TEXT,
۷۰۶     score        INT,
۷۰۷     CONSTRAINT fk_trip
۷۰۸         FOREIGN KEY (trip_number)
۷۰۹             REFERENCES trip (trip_id),
۷۱۰     CONSTRAINT fk_admin
۷۱۱         FOREIGN KEY (user_id)
۷۱۲             REFERENCES user_account (account_id)
);
۷۱۳
۷۱۴ CREATE TABLE hotel_survey
(
۷۱۵     survey_id    SERIAL not null PRIMARY KEY,
۷۱۶     hotel_id     INT,
۷۱۷     user_id      INT,
۷۱۸     comment      TEXT,
۷۱۹     score        INT,
۷۲۰     CONSTRAINT fk_hotel
۷۲۱         FOREIGN KEY (hotel_id)
۷۲۲             REFERENCES hotel (hotel_id),
۷۲۳     CONSTRAINT fk_admin
۷۲۴         FOREIGN KEY (user_id)
۷۲۵             REFERENCES user_account (account_id)
);
۷۲۶
۷۲۷ CREATE TABLE trip_ticket
(
۷۲۸     ticket_id     SERIAL not null PRIMARY KEY,
۷۲۹     trip_reservation_id INT,
۷۳۰     user_id       INT,
۷۳۱     examiner_id   INT,
۷۳۲     message       TEXT,
۷۳۳     type          TEXT,
۷۳۴     CONSTRAINT fk_trip
۷۳۵         FOREIGN KEY (trip_reservation_id)
۷۳۶             REFERENCES trip_reservation (reserve_id),
۷۳۷     CONSTRAINT fk_user
۷۳۸         FOREIGN KEY (user_id)
۷۳۹             REFERENCES user_account (account_id),
۷۴۰     CONSTRAINT fk_examiner
۷۴۱         FOREIGN KEY (examiner_id)
۷۴۲             REFERENCES user_account (account_id)
);
۷۴۳
۷۴۴ CREATE TABLE hotel_ticket
(
۷۴۵     ticket_id     SERIAL not null PRIMARY KEY,
۷۴۶     hotel_reservation_id INT,
۷۴۷     user_id       INT,
۷۴۸     examiner_id   INT,
۷۴۹     message       TEXT,
۷۵۰     type          TEXT,
۷۵۱     CONSTRAINT fk_hotel
۷۵۲         FOREIGN KEY (hotel_reservation_id)
۷۵۳             REFERENCES hotel_reservation (reserve_id),
۷۵۴     CONSTRAINT fk_user
۷۵۵         FOREIGN KEY (user_id)
۷۵۶             REFERENCES user_account (account_id),
۷۵۷     CONSTRAINT fk_examiner
۷۵۸         FOREIGN KEY (examiner_id)
۷۵۹             REFERENCES user_account (account_id)
);
۷۶۰
۷۶۱

```

```

۴۲۱ CREATE TABLE phone_call_hotel
۴۲۲ (
۴۲۳     phone_call_id SERIAL not null PRIMARY KEY,
۴۲۴     caller_id      INT    not null,
۴۲۵     callee_id     INT    not null,
۴۲۶     ticket_id     int     not null,
۴۲۷     duration       INTERVAL,
۴۲۸     date          DATE,
۴۲۹     time          TIME,
۴۳۰     CONSTRAINT fk_caller
۴۳۱         FOREIGN KEY (caller_id)
۴۳۲             REFERENCES user_account (account_id),
۴۳۳     CONSTRAINT fk_callee
۴۳۴         FOREIGN KEY (callee_id)
۴۳۵             REFERENCES user_account (account_id),
۴۳۶     CONSTRAINT fk_ticket
۴۳۷         FOREIGN KEY (ticket_id)
۴۳۸             REFERENCES hotel_ticket (ticket_id)
۴۳۹ );
۴۴۰
۴۴۱ CREATE TABLE phone_call_trip
۴۴۲ (
۴۴۳     phone_call_id SERIAL not null PRIMARY KEY,
۴۴۴     caller_id      INT    not null,
۴۴۵     callee_id     INT    not null,
۴۴۶     ticket_id     int     not null,
۴۴۷     duration       INTERVAL,
۴۴۸     date          DATE,
۴۴۹     time          TIME,
۴۵۰     CONSTRAINT fk_caller
۴۵۱         FOREIGN KEY (caller_id)
۴۵۲             REFERENCES user_account (account_id),
۴۵۳     CONSTRAINT fk_callee
۴۵۴         FOREIGN KEY (callee_id)
۴۵۵             REFERENCES user_account (account_id),
۴۵۶     CONSTRAINT fk_ticket
۴۵۷         FOREIGN KEY (ticket_id)
۴۵۸             REFERENCES trip_ticket (ticket_id)
۴۵۹ );
۴۶۰
۴۶۱
۴۶۲ CREATE TABLE ticket_type_priority
۴۶۳ (
۴۶۴     type      TEXT not null primary key,
۴۶۵     priority  INT
۴۶۶ );

```

ساخت index

```

1 CREATE INDEX idx_hotel_name ON hotel (name);
2 CREATE INDEX idx_user_name ON user_account (name);
3 CREATE INDEX idx_hotel_type ON hotel (type);
4 CREATE INDEX idx_company_score ON company (score);

```

برای هتل، علاوه بر ایندکس اصلی که id آن است، یک ایندکس هم بر اساس نام هتل‌ها گذاشتیم. این باعث تسريع در اجرای کوئری‌ها می‌شود. برای مثال اگر میخواهیم به دنبال نام یک هتل بگردیم، با این ایندکس راحت‌تر میتوانیم این کار را بکنیم.

برای هتل همچنین یک ایندکس دیگر برای type (تعداد ستاره‌ها) تعريف کردیم. اگر کاربری بخواهد بهترین هتل‌ها را ببیند، میتوانیم بر اساس این ایندکس به راحتی این هتل‌ها را به ترتیب خروجی دهیم.

برای شرکت‌ها هم بر اساس امتیازشان ایندکس گذاشتیم که اگر بخواهیم در جایی شرکت‌ها را با بالاترین امتیاز یا در رنج خاصی از امتیاز نمایش دهیم، به راحتی قابل انجام باشد.

برای کاربران هم یک ایندکس بر اساس اسمای آنها گذاشتیم. اگر بخواهیم اطلاعات یک کاربر را با داشتن نامش به دست آوریم، با استفاده از این ایندکس به راحتی امکان‌پذیر است.

با توجه به کوئری‌های بخش SQL باقی Index‌ها به این صورت ساخته شدند:

```

-- more indexes

CREATE INDEX idx_trip_start_date_vehicle_name ON trip (start_date, vehicle_name);

CREATE INDEX idx_vehicle_type_vehicle_name ON vehicle (type, vehicle_name);

CREATE INDEX idx_room_beds_number_room_id ON room (beds_number, room_id);

CREATE INDEX idx_hotel_amenity_name_hotel_id ON hotel_amenity (amenity_name, hotel_id);

CREATE INDEX idx_discount_code_type ON discount (code, type);

CREATE INDEX idx_discount_usage_trip_discount_code ON discount_usage_trip (discount_code);

CREATE INDEX idx_discount_usage_hotel_discount_code ON discount_usage_hotel (discount_code);

CREATE INDEX idx_trip_reservation_state_trip_number ON trip_reservation (state, trip_number);

CREATE INDEX idx_trip_start_date_vehicle_name ON trip (start_date, vehicle_name);

CREATE INDEX idx_trip_reservation_state_trip_number ON trip_reservation (state, trip_number);

CREATE INDEX idx_hotel_city_hotel_id ON hotel (city, hotel_id);

CREATE INDEX idx_trip_reservation_user_id_trip_number ON trip_reservation (user_id, trip_number);

CREATE INDEX idx_hotel_reservation_hotel_id_start_date ON hotel_reservation (hotel_id, start_date);

```

ساخت سرویس برای Rest API

برای زدن این بخش مجبور شدیم که با انواع ریکوئست‌های HTTP آشنا بشویم. در اینجا نیاز بود یک فریمورک برای پیاده‌سازی وب‌سرویس بک‌اندی خودمون انتخاب کنیم.

ما از زبان برنامه‌نویسی JS و فریمورک Fastify استفاده کردیم و وب‌سرویس را پیاده‌سازی کردیم.

همچنین برای زدن ریکوئست‌های Post از ابزار Postman استفاده کردیم که در ادامه به آن اشاره خواهیم کرد.

```

1 const fastify = require("fastify")();
2
3 fastify.register(require("@fastify/postgres"), {
4   connectionString: "postgres://postgres:1111@localhost:5432/db_project"
5 });

```

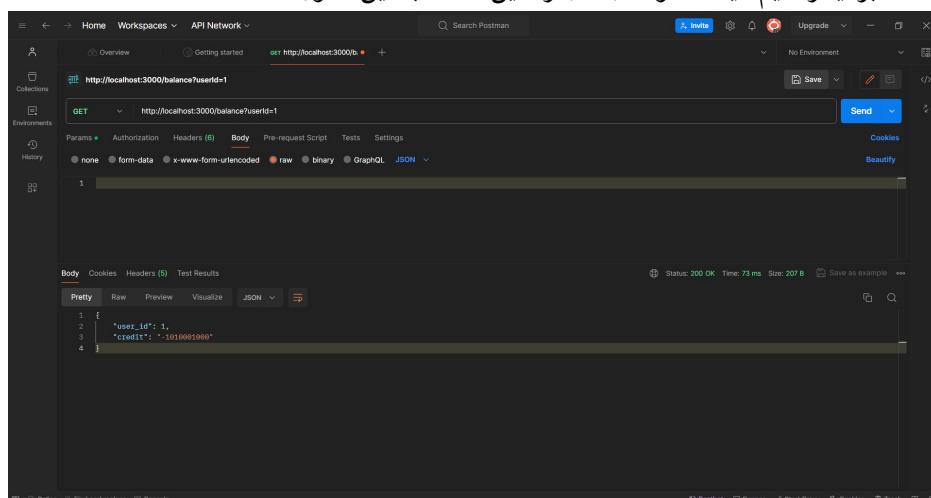
در اینجا کتابخانه Fastify را فراخوانی می‌کنیم و سپس با استفاده از آن به دیتابیس خود که روی localhost و پورت ۵۴۳۲ بالا است وصل می‌شویم.

```

1 fastify.get("/balance", async (req, res) => {
2   const userId = req.query?.userId;
3
4   if (!userId) {
5     return res.status(400).send("Please Enter userId ...");
6   }
7
8   const query = `
9     SELECT user_id, credit
10    FROM user_account
11   JOIN wallet ON wallet.user_id = user_account.account_id
12 WHERE user_account.account_id = $1
13 `;
14
15   try {
16     const result = await fastify.pg.query(query, [userId]);
17
18     if (result.rowCount === 0) {
19       return res.status(404).send("User not found");
20     }
21
22     res.send(result.rows[0]);
23   } catch (err) {
24     console.error("Database query error: ", err);
25     res.status(500).send("Internal Server Error");
26   }
27 });

```

در این بخش اولین ریکوئست ما هندل شده است. این ریکوئست از نوع Get است و با استفاده از userID ما میزان موجودی آن کاربر را به همراه Id برミگردانیم. یک نمونه از کارکرد این API به این صورت است:

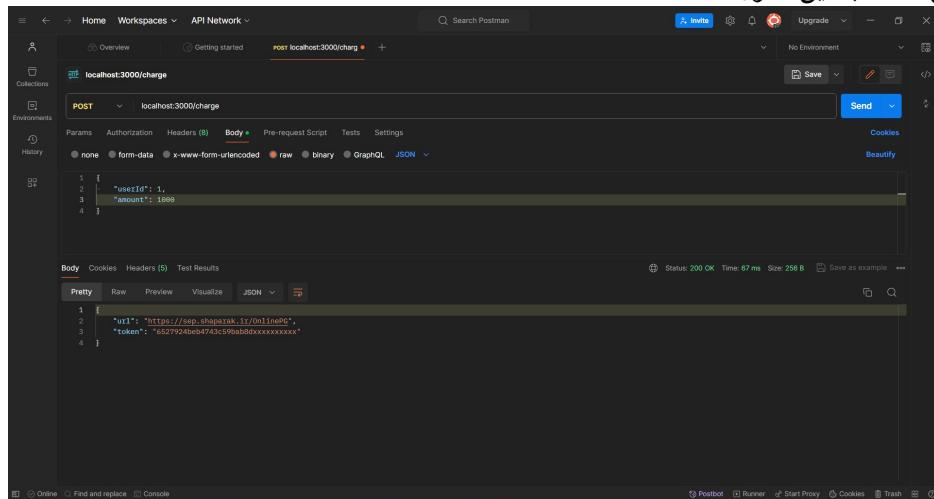


```

1 fastify.post("/charge", async (req, res) => {
2   const userId = req.body?.userId;
3   const amount = req.body?.amount;
4
5   const query = `
6     SELECT ACCOUNT_NUMBER
7       FROM bank_user
8      WHERE user_id = $1;
9
10  const query2 = `
11    INSERT INTO wallet_account_tr (token, bank_account_number, user_id, date, amount, type, state)
12    VALUES ('6527924beb4743c59bab8dxxxxxxxxx', $1, $2, '2024-06-01', $3, 'Deposit', 'Pending');
13
14  if (!(userId && amount)) res.send("Please enter userId and amount ...");
15  try {
16    let result = await fastify.pg.query(query, [userId])
17    const accountNumber = result.rows[0].account_number
18    result = await fastify.pg.query(query2, [accountNumber, userId, amount])
19    res.send({ "url": "https://sep.shaparak.ir/OnlinePG", "token": "6527924beb4743c59bab8dxxxxxxxxx"
20      });
21  } catch (error) {
22    console.error("Database query error: ", err);
23    res.status(500).send("Internal Server Error");
24  }
25});

```

ریکوئست این بخش از نوع Post است که تراکنش ایجاد کرده و لینک درگاه پرداخت به همراه یک توکن به وی برگردانده میشود. کارکرد این API به این صورت است:



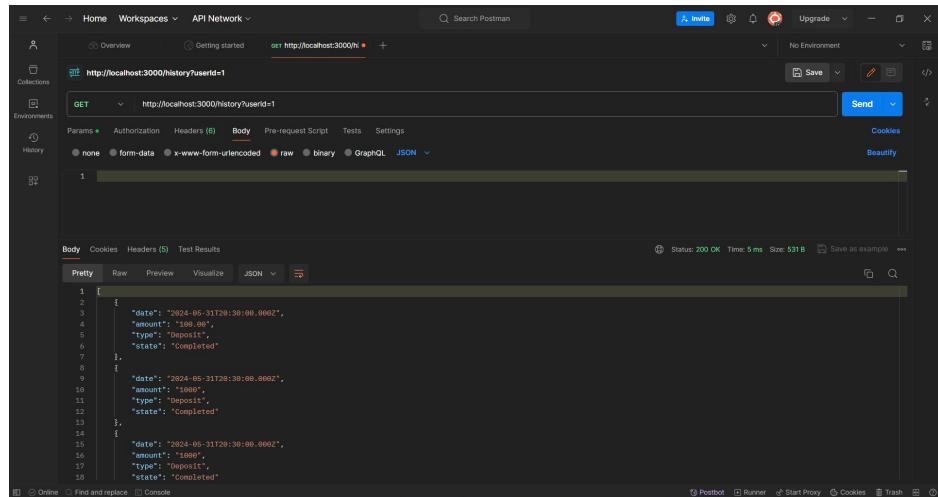
البته مقدار توکن را به صورت هاردکد قرار دادیم چون تولید یک توکن یکتا و ذخیره آن جزوی از سرویس ما نیست و باید داخل یک سرویس دیگر پیاده سازی شود و ما از آن استفاده کنیم.

```

1 fastify.post("/verify", async (req, res) => {
2   const userId = req.body?.userId;
3   const token = req.body?.token;
4   const amount = req.body?.amount;
5   if (!token) res.send("Please enter token ...");
6   const query = `
7     UPDATE wallet_account_tr
8     SET state = 'Completed'
9     WHERE token = $1 and user_id = $2;
10    `;
11
12   const query2 = `
13     SELECT credit
14     from wallet
15     where user_id = $1
16    `;
17
18   const query3 = `
19     UPDATE wallet
20     SET credit = $2
21     WHERE user_id = $1;
22    `;
23
24   try {
25     let result = await fastify.pg.query(query, [token, userId])
26     let credit_old = await fastify.pg.query(query2, [userId])
27     result = await fastify.pg.query(query3, [userId, credit_old.rows[0].credit + amount])
28     res.send({ "status": "verified" });
29   } catch (error) {
30     console.error("Database query error: ", err);
31     res.status(500).send("Internal Server Error");
32   }
33 });

```

این بخش همانند API قبلي از نوع Post است و باید یک ریکوئست بزنیم و تراکنش قبلی را تایید و ثبت کنیم. کارکرد این API به این صورت است:

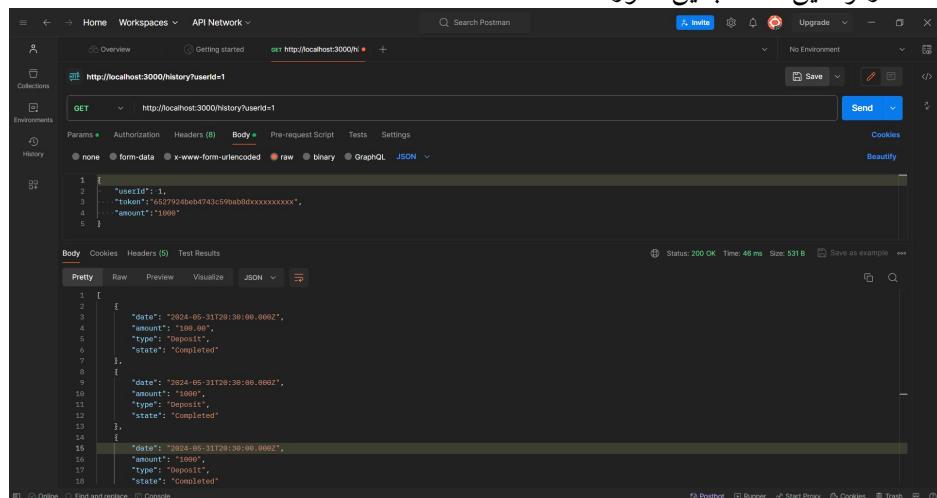


```

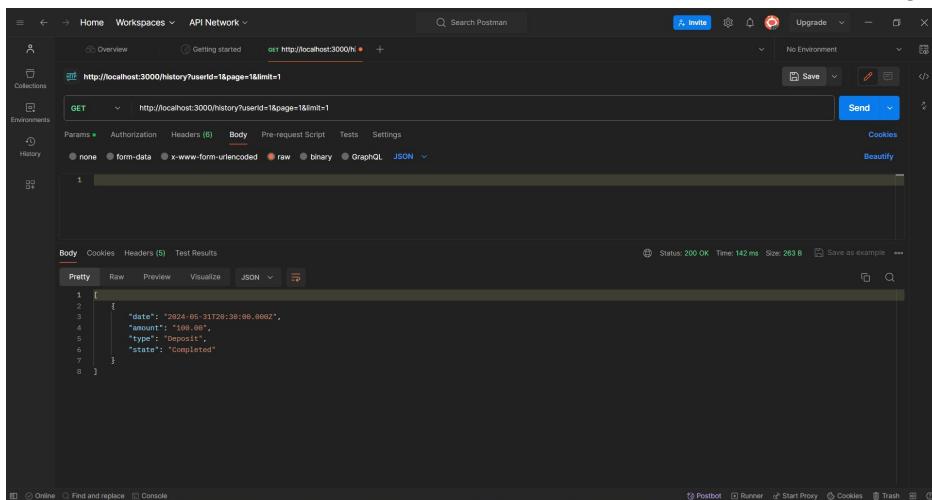
1 fastify.get("/history", async (req, res) => {
2   const userId = req.query?.userId;
3   if (!userId) return res.send("Please Enter userId ...");
4
5   const page = req.query?.page;
6   const limit = req.query?.limit;
7   const offset = (page - 1) * limit;
8
9   const query_offset = ` 
10    SELECT date, amount, type, state
11    FROM wallet_account_tr
12   WHERE user_id = $1
13   AND state = 'Completed'
14  ORDER BY date
15  LIMIT $2 OFFSET $3
16 `;
17
18   const query = ` 
19    SELECT date, amount, type, state
20    FROM wallet_account_tr
21   WHERE user_id = $1
22   AND state = 'Completed'
23  order by date
24 `;
25
26   if (page && limit) {
27     try {
28       const result = await fastify.pg.query(query_offset, [userId, limit, offset]);
29       res.send(result.rows);
30     } catch (err) {
31       console.error("Database query error: ", err);
32       res.status(500).send("Internal Server Error");
33     }
34   } else {
35     try {
36       const result = await fastify.pg.query(query, [userId]);
37       res.send(result.rows);
38     } catch (err) {
39       console.error("Database query error: ", err);
40       res.status(500).send("Internal Server Error");
41     }
42   }
43 }
44 );

```

در این بخش که آخرین API است یک ریکوئست Get داریم که به دو صورت می‌تواند باشد، یکی اینکه فقط با Rیکوئست را بزنند و یکی این که علاوه بر آن به ما page و limit هم بدهد تا کوئری ای بزنیم که مناسب در صفحات وب است. کارکرد این API بدین صورت است:



می بینید که همه تراکنش های موفق آن کاربر به ترتیب تاریخ برای ما برگردانده شده است. حال اگر ریکوئست را برای Pagi nation بزنیم، به این صورت است:



```

\ fastify.listen({ port: 3000, host: "0.0.0.0" }, (err) => {
  if (err) throw err;
  console.log(`server listening on ${fastify.server.address().port}`);
});

```

در آخر این وب سرویس را روی پورت ۳۰۰۰ اجرا کرده و ریکوئست ها را با استفاده از postman می زنیم.

همچنین در وب سرویس ما تا جای ممکن ارور را هندل کردیم، به عنوان مثال اگر یوزری پیدا نمی‌شود ارور مخصوص میدهیم:

The screenshot shows a POST request to `http://localhost:3000/balance?userId=22`. The response status is 404 Not Found, with the message "User not found".

یا اینکه اگر یک سری از فیلدها خالی باشد موقع ریکوئست زدن، ما ارور مناسب برمی‌گردانیم:

The screenshot shows a POST request to `http://localhost:3000/charge`. The response status is 200 OK, but it includes an error message: "Please enter userId and amount ...".

همچنین در همه جا از `try` و `catch` استفاده کردیم تا ارورهای احتمالی دیتابیس باعث توقف سرویس ننشود.