به نام خدا
پایگاه‌داده – ۱۴۰3

نام و نام خانوادگی: **شماره دانشجویی:**

مسیح تنورساز 40006133

سوال 1:

اسلایدهای شماره 47 و 48 مربوط به فصل 6.

Composite attributes: برای پیاده سازی این نوع از attribute ها، همه‌ی component attribute ها را یک attribute جداگانه در نظر میگیریم و آن‌ها را در جدول اضافه میکنیم. مثلا برای آدرس که یک Composite attributes است، component attribute های آن که street_name, apt_number, city, state, zip_code باشند را به عنوان attribute های جداگانه در جدول لحاظ میکنیم.

Multivalued attributes: برای پیاده سازی این نوع از attribute ها، یک جدول جداگانه در نظر میگیریم و رکورد های متناظر را در آن قرار میدهیم. مثلا برای استاد که ممکن است بیش از یک شماره داشته باشد، جدولی جداگانه تشکیل میدهیم، Attribute های این جدول شامل Id, phone_number اند. هر رکورد این جدول متناسب با استاد و شماره تلفن این استاد است. اسم این جدول را میتوان ترکیبی از استاد و شماره تلفن قرار داد. مثلا inst_phone.

سوال 2:

اسلایدهای شماره 38 الی 40 مربوط به فصل 6.

Weak Entity Set: Entity Set ای است که وجود آن نیازمند وجود موجودیت (موجودیت قوی) دیگر باشد. همچنین با توجه به attribute های موجود در این Entity Set، نمیتوان هر Entity را به صورت جداگانه و یا Unique مشخص کرد. دو مستطیل تو در تو نمایانگر این Entity Set است.

Strong Entity Set: هر Entity Set ای که Weak نباشد، Strong است. با یک مستطیل ساده آن را نشان میدهیم.

به عنوان مثال Person(ID, Name, ...) یک Strong Entity Set است و Student(school_name, degree, ...) یک Weak Entity Set است که برای وجود داشتن نیاز به Person دارد.

برای حذف موجودیت ضعیف دو راه داریم:

1. برای تبدیل Weak Entity Set به Strong Entity Set می‌توان Primary Key مرتبط با Strong را به Weak نیز اضافه کرد، اما در صورت انجام چنین کاری Redundancy افزایش می‌یابد و در نهایت ممکن است به ناسازگاری یا Inconsistency منتج شود.
2. همچنین می‌توان رابطه را حذف کرد و از شر آن خلاص شد که در این صورت موجودیت Weak یک مفهوم ضمنی و ناکامل را میرساند که مدنظر ما نیست. پس نمیتوان هیچ یک از دو راه را انجام داد.

بنابر این حتما نیاز به وجود Weak Entity Set ها داریم.

سوال 3:

برای تبدیل ERD موردنظر به یک پایگاه داده حداقل نیازمند **11** عدد جدول هستیم، چراکه:

1. برای **هر Entity** مانند (A, B, C, D) نیازمند یک جدول هستیم که در مجموع **4** جدول میشود.
2. برای **هر Relationship** مانند (R1, R2, R3, R4, R5) نیازمند یک جدول هستیم که در مجموع **5** جدول میشود.

3. همچنین برای **هر یک از Multivalued attribute** ها مانند (b2, d2) نیز نیازمند یک جدول هستیم که در مجموع **2** جدول میشود.

در نتیجه نیازمند **11** عدد جدول هستیم.

سوال 4:

الف) این نمودار به خواسته های زیر پاسخ میدهد:

1. وام دهی به مشتری ها
2. مشاهده اطلاعات مشتریان
3. تعریف حساب برای مشتری ها

ب) موجودیت Bank Branch ضعیف و بقیه‌ی موجودیت ها یعنی Bank, Loan, Account, Customer موجودیت های قوی هستند.

ج)

1) رابطه Branches:
1. این رابطه بین موجودیت‌های Bank, Bank Branch است.
2. یک رابطه یک به چند است.
3. هر شعبه مختص به یک بانک است.
4. هر بانک میتواند چندین شعبه داشته باشد.

2) رابطه Accts:
1. این رابطه بین موجودیت‌های Account, Bank Branch است.
2. یک رابطه یک به چند است.
3. هر اکانت مختص به یک شعبه است.
4. هر شعبه میتواند چندین اکانت داشته باشد.

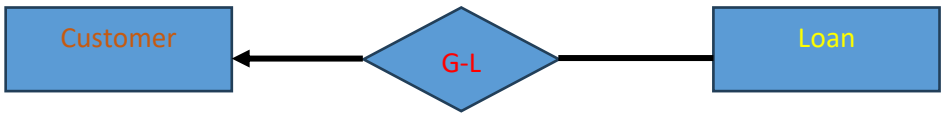
3) رابطه Loans:
1. این رابطه بین موجودیت‌های Loan, Bank Branch است.
2. یک رابطه یک به چند است.
3. هر وام مختص به یک شعبه است.
4. هر شعبه میتواند چندین وام داشته باشد.

4) رابطه A-C:
1. این رابطه بین موجودیت‌های Account, Customer است.
2. یک رابطه چند به چند است.
3. هر اکانت میتواند مختص به چند مشتری باشد.
4. هر مشتری میتواند چندین اکانت داشته باشد.

5) رابطه L-C:
1. این رابطه بین موجودیت‌های Loan, Customer است.
2. یک رابطه چند به چند است.
3. هر وام میتواند مختص به چند مشتری باشد.

4. هر مشتری میتواند چندین وام داشته باشد.

د) یک رابطه به اسم G-L را فرض میکنیم. این رابطه‌ی چند به یک، میان وام و مشتری است، به این مفهوم که هر وام تنها نیاز به یک مشتری برای ضمانت دارد و همچنین هر مشتری میتواند ضامن چندین وام باشد.(مشکل این کار این است که ضامن الزاما باید از میان مشتریان باشد)



راه دیگر تعریف یک موجودیت مستقل به نام ضامن است.

سوال 5:

سوال 6: