به نام خدا

گزارش پروژه: طراحی و پیاده‌سازی پایگاه داده سیستم مدیریت کلینیک

اعضای گروه:

- مهتا میرزایی 40131983
- زهرا ابوطالبی 40116943

تاریخ: ۳ خرداد ۱۴۰۴

---

۱. مقدمه (Introduction)

پروژه حاضر به طراحی و پیاده‌سازی یک پایگاه داده جامع برای سیستم مدیریت کلینیک اختصاص دارد. هدف اصلی این پروژه، ایجاد یک ساختار داده‌ای منسجم، کارآمد و قابل اطمینان برای مدیریت اطلاعات مربوط به بیماران، پزشکان، نوبت‌ها، نسخه‌های پزشکی و پرداخت‌ها در یک محیط کلینیک است. این سیستم با هدف بهبود فرآیندهای کاری، کاهش خطاهای دستی و ارائه دسترسی سریع و آسان به اطلاعات مورد نیاز کاربران مختلف (بیماران، پزشکان و مدیران کلینیک) طراحی شده است.

---

۲. نیازمندی‌های کاربران (User Requirements)

سیستم پایگاه داده با در نظر گرفتن نیازهای سه گروه اصلی کاربران طراحی و پیاده‌سازی شده است:

الف) بیماران:

- **امکان دریافت نوبت از پزشکان مختلف:** بیماران قادر خواهند بود لیست پزشکان و تخصص‌های آن‌ها را مشاهده کرده و برای دریافت نوبت با پزشک مورد نظر اقدام کنند.
- **مشاهده سوابق پزشکی و نسخه‌های تجویز شده:** دسترسی به تاریخچه بیماری‌ها، آلرژی‌ها و داروهای تجویز شده در ویزیت‌های قبلی فراهم شده است.
- **اطلاع از هزینه‌ها و پرداخت آنلاین:** امکان مشاهده هزینه نوبت‌ها و پیگیری وضعیت پرداخت آن‌ها (هرچند عملیات پرداخت آنلاین در لایه اپلیکیشن پیاده‌سازی می‌شود، زیرساخت دیتابیس برای آن فراهم است).

ب) پزشکان:

- **مشاهده لیست بیماران و سوابق پزشکی آن‌ها:** پزشکان می‌توانند به لیست بیماران خود و جزئیات سوابق پزشکی مربوط به هر بیمار دسترسی داشته باشند.
- **ثبت ویزیت‌های انجام شده و تجویز نسخه:** قابلیت ثبت جزئیات ویزیت و تجویز داروهای لازم برای بیماران پس از معاینه فراهم شده است.
- **مشاهده برنامه کاری و نوبت‌های تعیین شده:** پزشکان قادر خواهند بود برنامه کاری و نوبت‌های آتی و گذشته خود را مشاهده و مدیریت کنند.

ج) مدیریت کلینیک:

- **بررسی گزارش‌های آماری از تعداد ویزیت‌ها و درآمدها:** دسترسی به گزارش‌های تحلیلی و آماری برای ارزیابی عملکرد کلینیک و وضعیت مالی آن.

- **مدیریت پزشکان کلینیک:** قابلیت اضافه کردن، ویرایش اطلاعات و حذف پزشکان (با رعایت قوانین کسب‌وکار).
- **بررسی وضعیت مالی کلینیک و مدیریت پرداخت‌ها:** نظارت بر وضعیت پرداخت‌ها، شناسایی نوبت‌های پرداخت نشده و ثبت پرداخت‌های دستی.

---

۳. طراحی پایگاه داده (Database Design)

طراحی پایگاه داده بر اساس مدل رابطه‌ای (Relational Model) انجام شده است. تمرکز بر نرمال‌سازی داده‌ها، حفظ یکپارچگی ارجاعی (Referential Integrity) و بهینه‌سازی عملکرد بوده است. در طراحی پایگاه داده، اصول مهمی نظیر مدیریت فیلدهای مشتق، فیلدهای چند مقداری و روابط ضعیف مد نظر قرار گرفته‌اند.

نمودار ER (Entity-Relationship Diagram): نمودار ER سیستم در فایل ER.pdf ارائه شده است. این نمودار، موجودیت‌های اصلی سیستم (Patient, Doctor, Appointment, Prescription, Payment, Specialization, Patient_Medical_History) و روابط بین آن‌ها را به صورت گرافیکی نمایش می‌دهد.

معرفی جداول (Tables):

1. **Specialization:**
   - **هدف:** ذخیره اطلاعات مربوط به تخصص‌های مختلف پزشکی.
   - **فیلدها:** `Specialization_ID` (کلید اصلی), `Title` (نام تخصص), `Description` (توضیحات تخصص).
2. **Patient:**
   - **هدف:** نگهداری اطلاعات بیماران.
   - **فیلدها:** `Patient_ID` (کلید اصلی), `First_Name`, `Last_Name`, `Birth_Date`, `Phone_Number`.
3. **Doctor:**
   - **هدف:** نگهداری اطلاعات پزشکان.
   - **فیلدها:** `Doctor_ID` (کلید اصلی), `FirstName`, `LastName`, `PhoneNumber`, `Visit_Fee` (هزینه ویزیت), `Specialization_ID` (کلید خارجی به Specialization), `IsDeleted` (مدیریت حذف دکتر).
4. **Appointment:**
   - **هدف:** ثبت و مدیریت نوبت‌های بیماران با پزشکان.
   - **فیلدها:** `Appointment_ID` (کلید اصلی), `Appointment_DateTime` (تاریخ و زمان نوبت), `Patient_ID` (کلید خارجی به Patient), `Doctor_ID` (کلید خارجی به Doctor), `Cost` (هزینه نوبت), `Status` (وضعیت نوبت: Scheduled, Completed, Cancelled).
5. **Payment:**
   - **هدف:** ثبت جزئیات پرداخت‌های مربوط به نوبت‌ها.
   - **فیلدها:** `Payment_ID` (کلید اصلی), `Appointment_ID` (کلید خارجی به Appointment), `Payment_Method` (روش پرداخت), `Payment_Date` (تاریخ پرداخت) , `PaymentStatus`(وضعیت پرداخت: Paid, refunded)
6. **Prescription:**
   - **هدف:** نگهداری اطلاعات نسخه‌های تجویز شده توسط پزشکان.
   - **فیلدها:** `ID` (کلید اصلی), `Appointment_ID` (کلید خارجی به Appointment), `Medication_Name` (نام دارو), `Dosage` (دوز مصرفی).

7. **Patient_Medical_History:**
   - **هدف:** ثبت سوابق پزشکی بیماران.
   - **فیلدها:** `Patient_Medical_History_ID` (کلید اصلی), `Patient_ID` (کلید خارجی به Patient), `Condition_Title` (عنوان بیماری/حالت), `Condition_Description` (توضیحات).

---

۴. پیاده‌سازی اجزای پایگاه داده (Database Component Implementation)

برای افزایش کارایی، امنیت و سازماندهی کد، از توابع (Functions)، رویه‌های ذخیره شده (Stored Procedures)، تریگرها (Triggers) و ویوها (Views) استفاده شده است. برای هر نوع، حداقل ۳ مورد کاربردی و منطقی پیاده‌سازی شده است:

الف) توابع (Functions): توابع، قطعه کدهایی هستند که یک مقدار را برمی‌گردانند و می‌توانند در کوئری‌ها استفاده شوند.

1. **`GetPatientAge (@PatientID INT)`:**
   - **هدف:** محاسبه و بازگرداندن سن بیمار بر اساس `Birth_Date` او.
   - **کاربرد:** در گزارش‌ها یا نمایش اطلاعات بیمار در لایه اپلیکیشن، بدون نیاز به ذخیره مستقیم سن.
2. **`GetLastVisitDate (@PatientID INT)`:**
   - **هدف:** بازگرداندن تاریخ و زمان آخرین ویزیت (نوبت) یک بیمار. (آخرین نوبت با وضعیت completed)
   - **کاربرد:** در پروفایل بیمار یا پزشک برای مشاهده سابقه ملاقات.
3. **`IsAppointmentPaid (@AppointmentID INT)`:**
   - **هدف:** بررسی اینکه آیا یک نوبت مشخص پرداخت شده است یا خیر (بازگرداندن 1 برای پرداخت شده و 0 برای پرداخت نشده).
   - **کاربرد:** در گزارش‌های مالی، لیست نوبت‌های پرداخت نشده و در لایه اپلیکیشن برای نمایش وضعیت پرداخت به بیمار و مدیر.

ب) رویه‌های ذخیره شده (Stored Procedures): رویه‌های ذخیره شده، قطعه کدهای SQL هستند که عملیات پیچیده‌تری را انجام می‌دهند و می‌توانند پارامتر بپذیرند.

1. **`CreateAppointment (@Patient_ID INT, @Doctor_ID INT, @Appointment_DateTime DATETIME)`:**
   - **هدف:** ثبت یک نوبت جدید در سیستم.
   - **کاربرد:** توسط بیماران برای رزرو نوبت. این SP به صورت خودکار `Visit_Fee` پزشک را به عنوان `Cost` نوبت ثبت می‌کند.
2. **`RegisterPayment (@Appointment_ID INT, @Payment_Method NVARCHAR(50))`:**
   - **هدف:** ثبت جزئیات پرداخت برای یک نوبت مشخص.
   - **کاربرد:** توسط بیماران (پس از پرداخت آنلاین) یا کارکنان کلینیک (برای ثبت پرداخت‌های دستی).
3. **`InsertPrescription (@Appointment_ID INT, @Medications NVARCHAR(MAX))`:**
   - **هدف:** ثبت یک یا چند دارو به عنوان نسخه برای یک نوبت مشخص.
   - **کاربرد:** توسط پزشکان پس از ویزیت بیمار. این SP قادر است لیستی از داروها با دوزهایشان را از یک رشته ورودی تجزیه کرده و در جدول `Prescription` ثبت کند.
4. **`CompleteAppointment (@Appointment_ID INT)`:**

- **هدف:** به‌روزرسانی وضعیت یک نوبت به 'Completed' (انجام شده). درصورتی که پرداخت برای نوبت انجام نشده باشد یا کنسل شده باشد نمیتوان وضعیت نوبت را به 'Completed' تغییر داد.
- **کاربرد:** توسط پزشکان پس از اتمام ویزیت.

5. **`GetDoctorRevenueReport (@StartDate DATETIME, @EndDate DATETIME)`:**
   - **هدف:** ارائه گزارش درآمد و تعداد نوبت‌ها به تفکیک پزشک در یک بازه زمانی مشخص.
   - **کاربرد:** برای مدیریت کلینیک جهت بررسی عملکرد مالی پزشکان.
6. **`AddNewDoctor (@FirstName NVARCHAR(50), @LastName NVARCHAR(50), @PhoneNumber NVARCHAR(15), @Specialization_ID INT, @Visit_Fee DECIMAL(10,2))`:**
   - **هدف:** اضافه کردن یک پزشک جدید به سیستم.
   - **کاربرد:** توسط مدیریت کلینیک برای افزودن پرسنل جدید.
7. **`UpdateDoctorInfo (@Doctor_ID INT, @FirstName NVARCHAR(50) = NULL, ...)`:**
   - **هدف:** ویرایش اطلاعات پزشک موجود.
   - **کاربرد:** توسط مدیریت کلینیک برای به‌روزرسانی مشخصات پزشکان.
8. **`AddNewPatient (@FirstName NVARCHAR(50), @LastName NVARCHAR(50), ... )`:**
   - **هدف:** اضافه کردن یک بیمار جدید به سیستم.
   - **کاربرد:** افزودن بیمار به سیستم.
9. **`CancelOverdueScheduledAppointments`:**

- **هدف:** در صورتی که زمان نوبت گذشته و وضعیت آن 'completed' نشده وضعیت آن را به کنسل شده تغییر میدهد.
- **کاربرد:** کنسل کردن خودکار نوبت هایی که انجام نشده. میتوان این SP را به یک جاب داد تا در انتهای هر روز اجرا شود.

ج) تریگرها (Triggers): تریگرها، قطعه کدهایی هستند که به صورت خودکار در پاسخ به رویدادهای خاص (مانند INSERT, UPDATE, DELETE) روی یک جدول اجرا می‌شوند و به حفظ یکپارچگی و اعمال قوانین کسب‌وکار کمک می‌کنند.

1. **`trg_PreventDoctorDelete`:**
   - **هدف:** بجای حذف دکتر Isdeleted آن را 1 میکند و تمام نوبت های پیش روی آن دکتر را کنسل میکند.
   - **کاربرد:** حفظ یکپارچگی داده‌ها و جلوگیری از حذف تصادفی اطلاعات حیاتی.
2. **`trg_PreventOverlappingAppointments`:**
   - **هدف:** جلوگیری از ثبت نوبت‌های تداخلی برای یک پزشک (همزمان بودن دو نوبت برای یک پزشک).
   - **کاربرد:** اطمینان از صحت برنامه کاری پزشکان و جلوگیری از خطاهای رزرو.
3. **`trg_PreventAppointmentDeletion`:**
   - **هدف:** بجای حذف نوبت وضعیت آن را به کنسل شده تغییر میدهد.
   - **کاربرد:** از بین نرفتن داده ها و استفاده در گزارش ها و ...
4. **`trg_RefundCancelledAppointment`:**
   - **هدف:** در صورتی که نوبت کنسل بشه پرداخت را برگشت میدهد و وضعیت آن را به refunded تغییر میدهد.
   - **کاربرد:** افزایش اتوماسیون و اطمینان از همگام‌سازی وضعیت نوبت با وضعیت پرداخت.

د) ویوها (Views): ویوها، جداول مجازی هستند که خروجی یک کوئری را نمایش می‌دهند و می‌توانند برای ساده‌سازی دسترسی به داده‌ها و افزایش امنیت استفاده شوند.

1. **`DoctorsWithStats`:**
   - **هدف:** نمایش اطلاعات پزشکان به همراه آمار (میانگین هزینه ویزیت و تعداد بیماران منحصر به فرد).
   - **کاربرد:** برای بیماران جهت انتخاب پزشک و برای مدیریت جهت بررسی عملکرد پزشکان.
2. **`UnpaidAppointments`:**
   - **هدف:** لیست کردن تمام نوبت‌هایی که هنوز پرداختی برای آن‌ها ثبت نشده است.
   - **کاربرد:** برای مدیریت کلینیک و بخش مالی جهت پیگیری پرداخت‌های معوقه.
3. **`PatientMedicalSummary`:**
   - **هدف:** ارائه خلاصه‌ای از سوابق پزشکی هر بیمار (فهرست بیماری‌ها به صورت یکپارچه).
   - **کاربرد:** برای نمایش سریع تاریخچه پزشکی بیمار در پروفایل او یا برای پزشکان.
4. **`ClinicOverallStats`:**
   - **هدف:** نمایش آمار کلی کلینیک شامل تعداد کل نوبت‌ها، درآمد بالقوه و واقعی، و تعداد نوبت‌ها بر اساس وضعیت.
   - **کاربرد:** برای مدیریت کلینیک جهت ارزیابی کلی عملکرد.
5. **`AppointmentsWithPaymentStatus`:**
   - **هدف:** نمایش تمام نوبت‌ها به همراه نام بیمار و پزشک، هزینه، وضعیت نوبت و وضعیت پرداخت.
   - **کاربرد:** برای مدیریت کلینیک جهت نظارت جامع بر نوبت‌ها و وضعیت مالی آن‌ها.

---

۵. داده‌های نمونه (Sample Data)

برای هر یک از جداول `Specialization`, `Patient`, `Doctor`, `Appointment`, `Prescription`, `Patient_Medical_History` و `Payment`، حدود ۱۰ رکورد داده آزمایشی درج شده است. این داده‌ها برای تست قابلیت‌ها و اطمینان از عملکرد صحیح توابع، رویه‌ها، تریگرها و ویوها به کار رفته‌اند. تنوع در وضعیت نوبت‌ها (Scheduled, Completed, Cancelled) و وضعیت پرداخت‌ها نیز در نظر گرفته شده است تا سناریوهای مختلف پوشش داده شوند.

---

۶. نتیجه‌گیری (Conclusion)

در این پروژه، یک پایگاه داده قوی و منعطف برای سیستم مدیریت کلینیک طراحی و پیاده‌سازی شده است. با رعایت اصول نرمال‌سازی، استفاده از کلیدهای اصلی و خارجی، و پیاده‌سازی هوشمندانه توابع، رویه‌های ذخیره شده، تریگرها و ویوها، زیرساختی محکم برای پشتیبانی از نیازمندی‌های کاربران (بیماران، پزشکان و مدیران) فراهم شده است. این پایگاه داده قابلیت‌های لازم برای مدیریت نوبت‌ها، سوابق پزشکی، نسخه‌ها و پرداخت‌ها را ارائه می‌دهد و می‌تواند به عنوان هسته اصلی یک سیستم مدیریت کلینیک مدرن عمل کند.

قابلیت‌های آینده (اختیاری): برای توسعه‌های آتی می‌توان به افزودن ماژول‌های حسابداری پیشرفته‌تر، سیستم اطلاع‌رسانی پیامکی/ایمیلی برای نوبت‌ها، و ادغام با سیستم‌های بیمه اشاره کرد. همچنین، بهبود مدیریت خطا در SPها و افزودن ایندکس‌ها برای بهینه‌سازی عملکرد در مقیاس بزرگتر می‌تواند مد نظر قرار گیرد.