EyeGO

Website Database 구축

2020147001 김남윤

1. Database Requirement Analysis

저는 3 년여전에 망막박리를 직업 앓았던 사람으로써 아래의 네이버 카페인 '빛을 놓지 않는 사람들'을 이용하였습니다. 하지만 '네이버' 카페이기에 갖는 용이한 접근성을 제외하고는 환자들에게 있어서 여러모로 불편한 점이 많았습니다. 이 때문에 저는 그 당시에만 잠깐 이용하고 현재는 이용하지 않고 있습니다. 대신 저는 망막박리 환자들 더 나아가 망막박리에 걸릴 확률이 높은 위험군과 관련 병 연구자, 의사 등에 도움이 되는 사이트를 제작하고자 했습니다. 그 이유는 망막박리라는 질병이 희소하고 지속적인 관리가 필요하기 때문입니다. 그렇기에 기존 카페와 달리 '정확한 자료'를 제공하며, 환자들의 지속적인 의료자료 기록을 통해 향후 망막박리에 대한 연구와 수술 후 관리에 '지속적인 도움'을 제공하는 것을 사이트의 목적으로 두었습니다.

물론, 의료 데이터라는 것이 환자들의 개인정보와 의사들의 협조 등이 필요하고 사이트 제작 후에 어느 정도의 데이터가 쌓여야 서비스 제공이 가능합니다. 따라서 이번의 DBMS 프로젝트에서는 망막박리와 관련 질병에 대한 정보를 수집하여 임의의 데이터를 '직접 제작'하였기에, 머신러닝이나 딥러닝 및 이미지처리의 application 기능 구현은 불가했습니다. 대신 앞의 임의 데이터를 통해 생성한 Operational DB 에서 Star Scheme 가 아닌 독립적인 Dimesional DB(Data Warehouse)를 제작하여 사이트의 핵심적인 기능 중 하나인 '환자들의 데이터를 통한 의미있는 통계자료 분석' 기능을 제공하는 것에 초점을 맞추었습니다. 현재 네이터 카페의 경우에는 카페 내에 저장된 자료를 통해서 오직 '기간, 게시판 선택 후 키워드 검색 기능'만을 제공함으로, 이 점을 보충하는 것이 이번 프로젝트의 목적이라고 할 수 있겠습니다.

추가적으로, 기존 데이터베이스를 통한 통계자료 분석 이외에 기존 네이버카페가 가지고 있는 기능인 '추천글'을 구현하고자 하였고, 망막박리 환자들만이 아닌 의사들과 연구자들을 위한 DB 를 추가로 제작하였습니다.

2. ER Model(EER) & Relational Database Design(RDB)

기존의 네이버 카페와 다른 새로운 사이트를 직접 제작하고자 하기에 ER Model 제작 시, 기존 네이버 카페와의는 차이가 있습니다. 먼저 자희 사이트는 환자, 의사 그리고 연구자로 사이트 방문자를 분류합니다. 그리고 데이터를 분석하기 위해 의료데이터와 수술 관련 데이터를 따로 분류하여 저장하고 분류합니다. 마지막으로 환자가 질문하면 의사가 답변해주는 Q&A 데이터와 사이트 참여자들이 모여 새로운 연구를 진행하도록 하는 연구 리스트 데이터도 따로 저장합니다.

(1) 각 테이블에 대한 설명

각 참여자들의 정보는 PATIENT, DOCTOR, RESEARCHER 테이블로 나타나며, 의료데이터에 대한 정보는 MEDICAL_DATA, 치료 및 수술에 대한 정보는 TREATMENT 테이블에 나타나 있습니다. Q&A 데이터의 경우는 QUESTION, ANSWER 테이블 그리고 프로젝트에 대한 정보는 PROJECT 테이블에 저장되어 있습니다.

(2) 테이블 간의 관계에 대한 설명 및 RDB 제작 과정

PATIENT-MEDICAL_DATA 경우에는 병원에 주기적인 방문을 하여 시간에 따른 데이터를 계속해서 생산해내기에 PATIENT 와 MEDICAL_DATA 는 1:N 의 Maximum Cardinalrity 를 갖습니다. 또한,

의료 데이터는 환자에 대한 정보가 있어야 그 자체로 의미가 있고(Identifying Relationship), 환자의 경우에도 회원가입시 반드시 처음 자신의 질료 기록을 입력하도록 하는 방침에 따라 M-M 의 Minimum Cardinalrity 를 갖습니다.

MEDICAL_DATA-TREATMENT 경우에는 환자가 병원을 방문했을시에 여러 수술 및 시술 기법(ex. 망막동률술, 레이져치료 등)들이 한 번에 이뤄지고, 한가지 치료라도 다양한 날에 걸쳐서 이뤄지는 수술이 있을 수 있고 재발에 대한 또 같은 치료가 이뤄질 수 있고, 다른 환자도 같은 수술을 받을 수 있기에 N:M 의 Maximum Cardinalrity 를 갖습니다. 그리고 치료의 경우에는 환자들이 여태껏 수술받아온 적이 있는 데이터를 기준으로 제작하였으며, 환자의 경우 정기적인 검진으로 병원에 방문해서 수술이나 시술을 받지 않을 수 있기에 M-O 의 Minimum Cardinalrity 를 갖습니다. 또한, 이전과 달리 Strong 한 관계이므로 Intersection Table 을 만들어주었습니다.

TREATMENT_DOCTOR 경우에는 의사가 여러 수술이나 치료를 할 수 있고, 한 수술이라도 할 수 있는 의사가 여러 명인 것이 보편적이기에 N:M 의 Maximum Cardinalrity 를 갖습니다. 그리고 의사의 경우에는 한 가지라도 할 수 있는 치료가 있다고 보편적으로 가정하고 치료의 경우에도 그 치료를 할 수 있는 한 명이라도 있기에 치료가 가능하다고 생각하여 M-M 의 Minimum Cardinalrity 를 갖습니다. 위와 마찬가지로 Strong 에 N:M 이기에 Intersection Table 을 만들어주었습니다.

MEDICAL_DATA-DOCTOR 경우 입니다. 잘 생각해보니 환자가 어떤 치료를 받았는지는 MEDICAL_DATA 와 TREATMENT 를 통해 알 수 있는데, 어떤 의사에게 치료받았는지를 알 수 없다는 것을 발견하였습니다. 병이 지속적인 관리가 필요하기에 환자의 경우, 보통 자신이 처음 진료받았던 의사 선생님을 찾아가서 병의 진행과정에 대한 믿음을 갖습니다. 따라서 어떤 의사가 나를 치료했는지에 대한 정보가 매우 중요합니다. 단, 여기서는 모든 의사들이 이 웹사이트에 가입을 했다는 조건하에서 입니다. 그렇게 진행해나가면, 한 환자의 경우 갑자기 의사가 마음에 들지 않아 다른 의사를 찾아갈 수 있고 의사의 경우에는 다양한 환자를 진찰하거나 같은 환자라도 날짜별로 받아들이기에 N:M 의 Maximum Cardinality 를 갖습니다. 또한, 의사의 경우에는 직업이 의사인 이상 무조건 1 명 이상의 환자가 존재하고 그 환자가 사이트에 가입을 한다는 보편적인 가정을 추가하고 환자도 수술 혹은 시술이라도 의사를 만나러 가기에 M-M 의 Maximum Cardinality 를 갖습니다. 위와 마찬가지로 Strong 에 N:M 이기에 Intersection Table 을 만들어주었습니다.

PATIENT-QUESTION 경우에는 환자가 여러 질문을 할 수 있고 한 질문은 반드시 한 명의 환자가 한 것이므로 1:M 의 Maximum Cardinality 를 갖습니다. 또한 질문을 안하는 환자가 있을 수 있고, 질문이 있다는 것은 그 환자가 탈퇴하지 않는 이상 질문자인 환자가 존재함으로 M:O 의 Minimum Cardinalrity 를 갖습니다.

QUESTION-ANSWER 경우에는 한 질문에 대해 여러 답변이 달릴 수 있지만, 한 답변은 여러 질문에 대한 답이 되는 것은 아니기에 1:N 의 Maximum Cardinalrity 를 갖습니다. 그리고 답변은 질문이 있어야 가능하지만, 질문에 대해 아직 답변이 하나도 없을 수 있기에 M-O 의 Minimum Cardinalrity 를 갖습니다. 여기서 답변이 질문에 대해 ID Dependent 가 아니지만, Weak 로 설정하여 답안 자체는 그 자체로 의미를 갖지만 질문이 사라지면, 같이 사라지도록 하였습니다.

ANSWER-DOCTOR 경우에는 환자가 질문을 한다면, 의사가 답변을 해주기에 이러한 관계를 가집니다. 각 답변은 해준 의사가 한 명이지만, 의사의 경우에는 여러 답변을 할 수 있기에 N:1 의 Maximum Cardinalrity 를 갖습니다. 답변이 달리면, 무조건 답변을 한 의사가 존재하지만 답변을 하나도 작성하지 않은 사이트 질문게시판에 비협조적인 의사가 존재할 수 있기에 O-M 의 Minimum Cardinalrity 를 갖습니다.

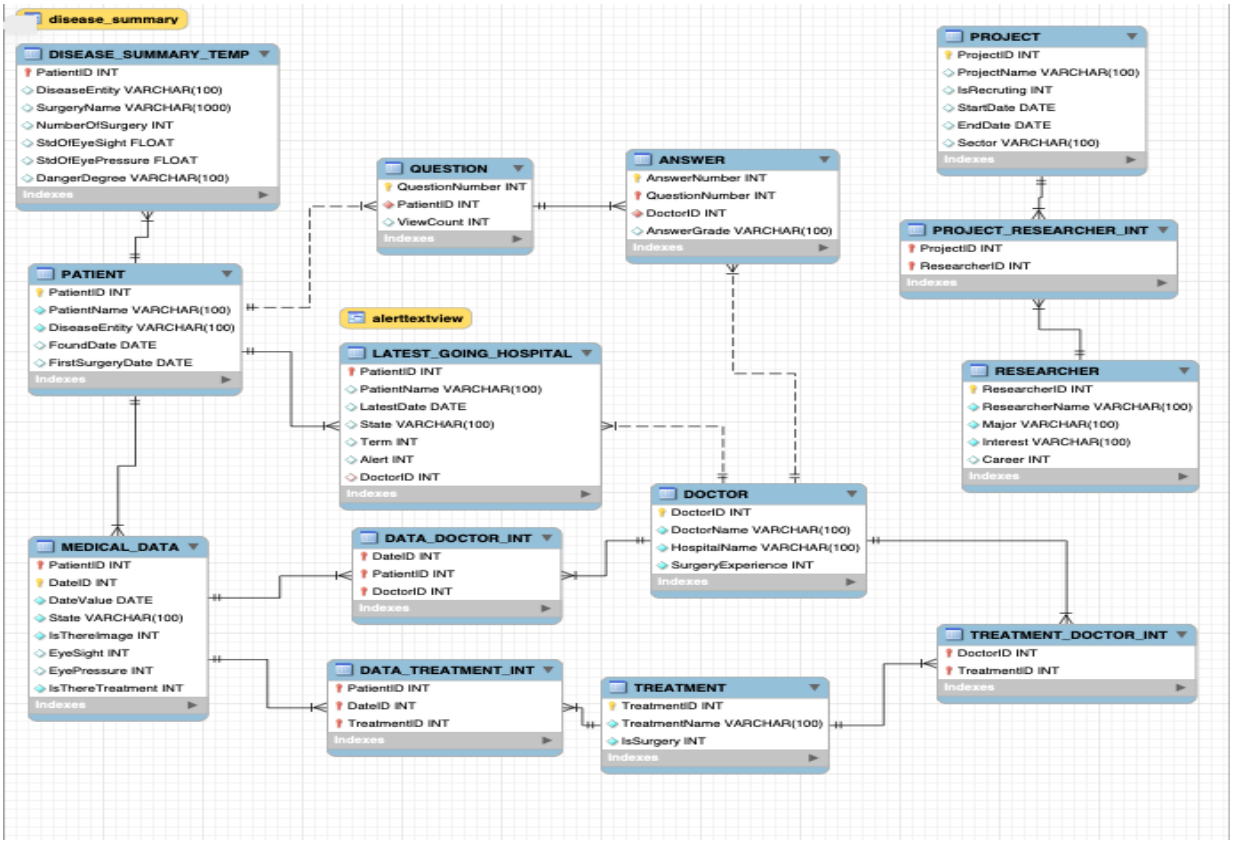
마지막으로 PROJECT-RESEARCHER 경우에는 한 프로젝트에 여러 연구자가 참여할 수도 있고, 여러 연구에 참여중인 열정있는 연구자가 있거나 혹은 종료된 연구까지 포함됨으로 한 연구자가 여러

프로젝트에 참여할 수도 있습니다. 그렇기에 N:M 의 Maximum Cardinalrity 를 갖습니다. 연구자의 경우, 새로 가입하여 프로젝트에 참여한 경력이 아직 없을 수도 있지만 프로젝트의 경우에는 반드시 1 명 이상의 연구자를 필요로 하기에 O-M 의 Minimum Cardinalrity 를 갖습니다. 이전과 마찬가지로 Strong 에 N:M 이기에 Intersection Table 을 만들어주었습니다.

추가로 PROJECT 의 경우에는 연구자뿐 아니라 환자와 의사까지 참여할 수 있도록 하는 것이기에 RPOJECT-PATIENT, PROJECT-DOCTOR 관계가 추가되어야 하지만 이전 PROJECT-RESEARCHER 과 같은 Relationship 을 갖으며 그렇기에 Intersection Table 이 복잡하게 추가되어야 함으로 여기서는 언급만 하는 것으로 끝내도록 하고자 합니다.

모든 Relationship 을 살펴보면, One 이면 반드시 Mandatory 으로 나타나게 됩니다. 즉, One-Optional 같은 가벼운 관계는 이 RDB 내에서는 존재하지 않습니다. 따라서 FK 를 가진 모든 곳에 NOT NULL 이라는 조건을 추가하여 Mandatory 를 적용할 수 있습니다. 뿐만 아니라 CASCADE UPDATE, CASCADE DELETE 를 모두 ACTION 으로 해두어 한 곳에서 지워지면, 다른 곳에서도 같이 지워질 수 있도록 하였습니다.

다만 아쉬운 점은, TREATMENT-DATA_TREATMENT_INT, PATIENT-QUESTION, QUESTION-ANSWER, ANSWER-DOCTOR, PROJECT-RESEARCHER 의 경우에는 Optional 의 관계가 있지만, Reverse Engeering 으로 EER 그림을 만들었기에 모두 Mandatory 로 표현되었습니다. 구체적인 것은 앞의 설명을 참고하면 될 듯합니다. 또한, Minimum Cardinality 의 경우에 일부는 NOT NULL, Referential Integrity(CASCADE UPDATE, CASCADE DELETE)로 구현했지만, 일부는 Trigger 를 통해 구현해야 한다는 과제를 남겨두고 있습니다. 이에 더하여 MEDICAL-DATA 의 IsThereTreatment 가 1 이어야 TREATMENT 와 연결된다는 사항 등을 비롯하여 테이블 간의 Relationship 으로 아직 표현이 불가한 사항들에 대한 코드 구현이 아직 남아 있습니다.



3. Application

이번 프로젝트에서는 2 가지의 기능을 MySQL 내의 기능인 Stored Procedure 기능을 사용하여 구현해 보았습니다. Stored Procedure 을 통해 Operation DB 로 제작한 RDB 를 떼어내어 각각을 Dimesional DB(Data Warehouse) 테이블로 제작하였고, 이를 토대로 의미있는 자료만 다시 골라서 뽑아낸 view 로 구현하는 절차를 밟았습니다.

(1) 정기적인 병원 방문 날짜 및 정보를 알려주는 기능

Dimesnional DB : *LATEST_GOING_HOSPITAL*
Stored Procedure : *InsertTermAndAlert*
View : *AlertTextView*

첫번째 기능은 병원 방문 날짜가 임박했음을 알려주는 기능입니다. 망막박리라는 병의 특성상 6 개월에 1 번씩 정기검진을 받아야합니다. 따라서 마지막 진료날짜에서부터 오늘까지 6 개월(180 일)이 지났을 경우에는 경고(Alert 0 -> 1)열의 정보가 바뀌는 것이 주요 기능입니다. View 를 통해 바로 구현하기가 어려워서 우선 Operational DB 를 통해 Dimensional DB 를 제작하였습니다. DB 테이블에는 환자의 정보, 마지막 방문 날짜, 마지막 방문했을 시의 상태, 마지막 방문 날짜로부터 지금까지 얼마나 지났는지, 그리고 6 개월이 지나면 경고 표시, 의사의 정보를 Stored Procedure 을 통해 집어넣었습니다. 추가로 6 개월이 지나지 않았더라고 3 개월(90 일)이 지났는데, 마지막 환자의 상태가 'Moderate'이상으로 상태가 좋지 않았던 경우에는 마찬가지로 경고(Alert 0 -> 1)열의 정보가 바뀌도록 하였습니다. 그리고 이러한 정보를 토대로 View 를 제작하였습니다. View 를 통해서 병원 가리는 경고가 뜬 사람들의 이름, 마지막으로 진단받은 의사이름, 의사가 속한 병원의 이름을 보여주도록하여 환자가 빨리 담당 의사와 병원에 연락하여 예약 및 진료를 할 수 있도록하였습니다. 아래의 사진은 Dimensional DB(왼쪽)과 View(오른쪽)의 모습입니다.

PatientID	PatientName	LatestDate	State	Term	Alert	DoctorID
1	Lob	2021-04-14	Weak	260	1	1
2	Bob	2021-09-16	Weak	105	0	2
3	John	2021-05-17	Moderate	227	1	2
4	Chris	2021-09-21	Moderate	100	1	4
5	Robert	2021-12-30	Weak	0	0	5

Patient	Doctor	Hospital
Lob	Dohun	Yonsei
John	Mikeal	Samsung
Chris	Christina	Asan

(2) 환자의 진료기록을 토대로 병에 대한 정보(관련 수술, 수술 후 후유증 및 사후 관리)

Dimesnional DB : *DISEASE_SUMMARY_TEMP*
Stored Procedure : *InsertNOPAndDD*
View : *DiseaseSummary*

두번째 기능은 환자의 진료기록을 토대로 병에 대한 정보를 추출해내는 것입니다. 이를 통해 환자와 연구자 그리고 의사들의 병에 대한 이해를 도울 수 있습니다. 특히, 환자의 경우에는 자신이 어떤 병에 걸렸을 경우, 지금까지 그 병에 걸린 환자 수는 얼마나되고, 받아야될 수술의 종류는 무엇이 있고, 평균적인 수술 횟수는 어느정도 되고, 수술전후로 시력의 변화는 어떻게 되며, 수술전후로 안압의 변화 또한 어떻게 되는지를 제공하는 기능입니다. 이전과 마찬가지로 바로 View 로 만들기에는 복잡한 과정을 처리해야하기에 Operational DB 에서 데이터를 추출한 Dimensional DB(Data Warehouse)를 제작하였습니다. 이 테이블은 환자별로 걸린 질병 이름, 지금까지 받은 수술의 이름, 지금까지 받은 수술의 수, 수술 전후로 시력 변화, 수술 전후로 안압변화, 그리고 이를 종합한 병의 위험 정도

수치입니다. 추가로 StdOfEyeSight 의 경우에는 정상 시력이라는 것이 존재하지 않기에 수술 전후로의 시력변화를 표준편차로 나타내었고, StdOfEyePressure 은 정상 안압 15 와의 절댓값의 총합으로 계산하였습니다. 마지막에 DangerDegree 의 경우에는 앞선 4 개의 열을 토대로 각각 0 점에서 1 점의 점수 범위로 산정하여 총 0 에서 4 점 만점의 위험도를 표기하려고 했으나, MySQL 상에서 구현하기에는 어려움이 있어서 보류상태로 남겨두었습니다. 구체적으로는 REPEAT~UNTIL 안에 이중 IF 문에서 오류가 발생하였습니다. Stored Procedure 의 주석 부분을 보시면, 구현은 되지 않았지만 어떤 형식으로 DangerDegree 를 수치화했는지 알 수 있으실 겁니다. 마지막으로 환자들을 비롯한 의사, 연구자가 의미있는 데이터를 손쉽게 볼 수 있도록 View 로 제작해보았습니다. 앞서 말했듯이 자신이 어떤 병에 걸렸을 경우, 지금까지 그 병에 걸린 환자 수는 얼마나되고, 받아야될 수술의 종류는 무엇이 있고, 평균적인 수술 횟수는 어느정도 되고, 수술전후로 시력의 변화는 어떻게 되며, 수술전후로 안압의 변화 또한 어떻게 되는지를 제공합니다. 아래의 사진은 Dimensional DB(위)과 View(아래의 모습입니다.

P	DiseaseEntity	SurgeryName	NumberOfSurge...	StdOfEyeSight	StdOfEyePressure	DangerDegree
1	Rhegmatogenous	Scleral buckling / Vitrectomy	2	0.4	2	NA
2	Rhegmatogenous	NULL	0	0	0	NA
3	Tractional	Pneumatic Retinopexy / Retinal Cryopexy / Scle…	5	0	9	NA
4	Tractional	Retinal Cryopexy / Scleral buckling / Vitrectomy	3	0.8	10	NA
5	Exudative	NULL	0	0	0	NA

Disease	PatientCount	Surgery	SurgeryCount	EyeSightDiff	EyePressureDiff
Exudative	1	NULL	0.0000	0	0
Rhegmatogenous	2	Scleral buckling / Vitrectomy	1.0000	0.20000000298023224	1
Tractional	2	Pneumatic Retinopexy / Retinal Cryopexy / Scle…	4.0000	0.4000000059604645	9.5

(3) 추후 과제

추가적으로 시간이 가능하다면, 구현할 기능에 대해 설명하도록 하겠습니다.

첫번째로, 환자들에게 의사에 대한 정보를 제공하는 기능입니다. 앞서 했던 것처럼 Dimensional DB(테이블) 제작, Stored Procedure 제작, 그리고 View 제작의 일련의 과정을 거친 뒤에 마지막으로 환자가 의사의 ID 를 함수에 넣어서 전달하면, View 나 앞서 제작한 테이블에서 정보를 추출하여 의사 이름, 지금까지 해온 수술 및 시술 종류, 받은 환자수, 호전도(시력변화나 압력변화로 수치화), 주로 치료한 병명 종류, 받은 환자수, 호전도(앞과 마찬가지) 등의 정보를 제공합니다.

두번째로, Q&A 게이판에 대한 정보를 알려주는 것입니다. 초반에 말했듯이 QUESTION 의 ViewCount 가 높은 게시글이나 ANSWER 의 AnswerGrade 가 높은 게시글이 인기게시글 및 추천게시글이 되어 view 로 게시글에 대한 정보(ID, 답변수 등)를 제공합니다. 이에 더하여 ViewCount 의 총 합이 높은 사람을 우수회원으로 선정하고, AnswerGrade 가 높은 혹은 자주 답변을 작성한 의사를 우수의사로 선정하는 기능도 사이트의 질문게시판을 원활하게 하기 위한 기능으로 사료됩니다.

마지막으로 아쉬운 것은 다른 사이트와 아주 차별화되는 기능인데, 아직 구현하지 못한 PROJECT 테이블에 관한 것입니다. 물론, 이전에 진행한 환자들이 제공한 정보를 통한 의미있는 자료 제공도 괜찮긴 하지만, PROJECT 기능이야 말로, 크게 DB 분석으로는 의미있는 결과를 뽑아내지는 못하지만 환자들을 비롯한 의사, 연구자들이 PROJECT 에 참여하여 희귀병이자 지속적인 관리가 필요한 이 질병에 대한 이해를 높여갈 수 있기에 반드시 필요한 기능이라고 생각합니다.

4. Other Application

앞선 MySQL 을 이용한 것뿐 아니라 머신러닝과 딥러닝 등을 이용하려면, 환자들의 진짜 데이터와 많은 데이터가 필요합니다. 그렇기에 웹사이트를 직접 제작하여 환자들을 모으고 DB 관리를 하는 것을 가장 우선적으로 해야한다고 생각합니다. 아래는 그 첫번째 단계인 웹프론트엔드에 대한 간단한 구상입니다. 완성도 있는 웹 프론트 제작이후, 파이썬과 SQL 을 이용한 DB 관리 후, 실제 상용화를 이뤄내면 앞선 ML, DL 를 할 수 있을 것으로 기대됩니다.