Implementação de um cursor simples em Python3

Guilherme Andreúce - 14/0141961
Gabriel Taumaturgo - 14/0140522
Jadiel Teófilo - 14/0022911
Thiago Luis Pinho - 15/0065205

8 de Dezembro de 2017

Resumo

O trabalho apresentado a seguir têm objetivo de mostrar ao leitor como utilizar cursores para percorrimento dos regristros de um banco de dados em Python3 com a ajuda de Selects e Views, bem como todo o processo de modelar o banco utilizando técnicas como a criação do Diagrama de Entidade e Relacionamento (DER), Modelo Relacional (MR), Normalização, processo de Extract, Transform and Load (ETL) e criação de Triggers e Procedures.

1 Introdução

Sabe-se que a utilização de Banco de Dados hoje em dia é essencial. Visto sua aplicabilidade em diversas áreas, o trabalho apresentado a seguir apresenta técnicas essenciais para a modelagem de um banco de dados funcional. Será abordado como construir um DER, MER, normalizar, realizar o ETL e, com o auxilio do cursor Connector Python, realizar consultas e manipular os registros do Banco já construído. Serão utilizados os microdados disponibilizados pelo INEP à respeito do Enem de 2015.

2 Modelando o Banco de Dados

Um modelo em banco de dados é o processo de se pegar o modelo entidade relacionamento ou modelo conceitual e transformá-lo em algo que seja fácil de se implementar computacionalmente. Assim surgem duas abordagens muito utilizadas para a construção do banco de dados que são o Diagrama Entidade Relacionamento ou DER e o Modelo Relacional ou MR. O DER é uma representação de alto nível do MR.

2.1 Diagrama Entidade Relacionamento

O DER utiliza de entidades, atributos e explica como as entidades se relacionam. Uma entidade é um elemento do mundo real com uma existência própria. Cada entidade possui propriedades que a descreve, chamadas de atributos. Relacionamento é um conjunto de associações entre entidades. Foi utilizado o programa EDRPlus para a construção do DER.

2.2 Modelo Relacional

O Modelo Relacional ou MR, foi inicialmente introduzido por Codd (1977) e representa os dados em um banco de dados como uma coleção de relações (tabelas). Cada linha é denominada tupla; O nome de uma coluna é chamado de atributo; A tabela é chamada de relação. Os atributos selecionados para constituir a chave de um relação são comumente denominados de chave primária da relação ou Primary Key ou PK.Uma chave estrangeira ou Foreign key ou FK é uma coluna ou uma combinação de colunas, cujos valores fazem referencia a chave primária de uma outra tabela. A chave estrangeira é o mecanimos que permite a implementação de um relacionamento em um banco relacional. Foi utilizado o programa MySQLWorkbench para a construção do MR.

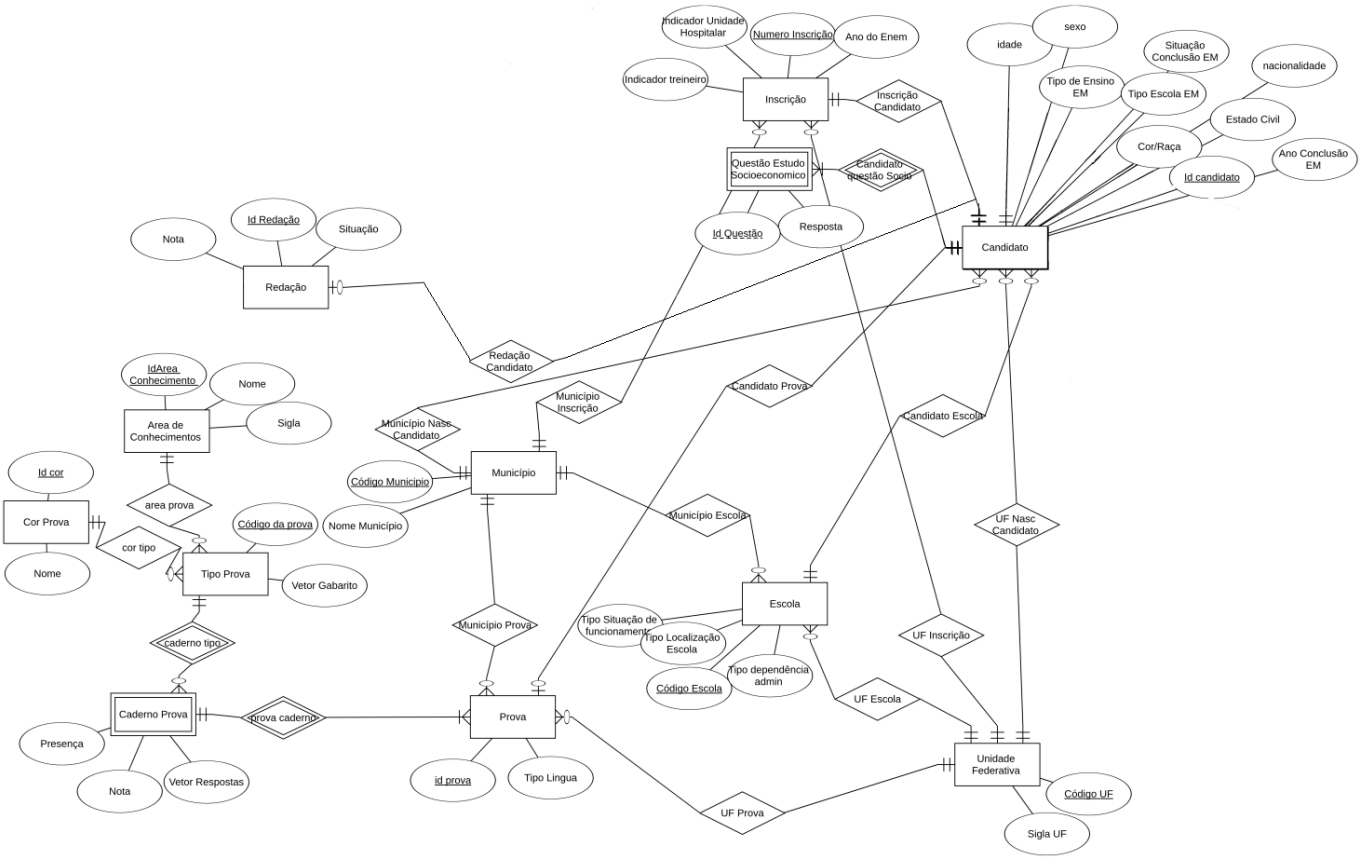

Figura 1: Diagrama Entidade Relacionamento Enem 2015

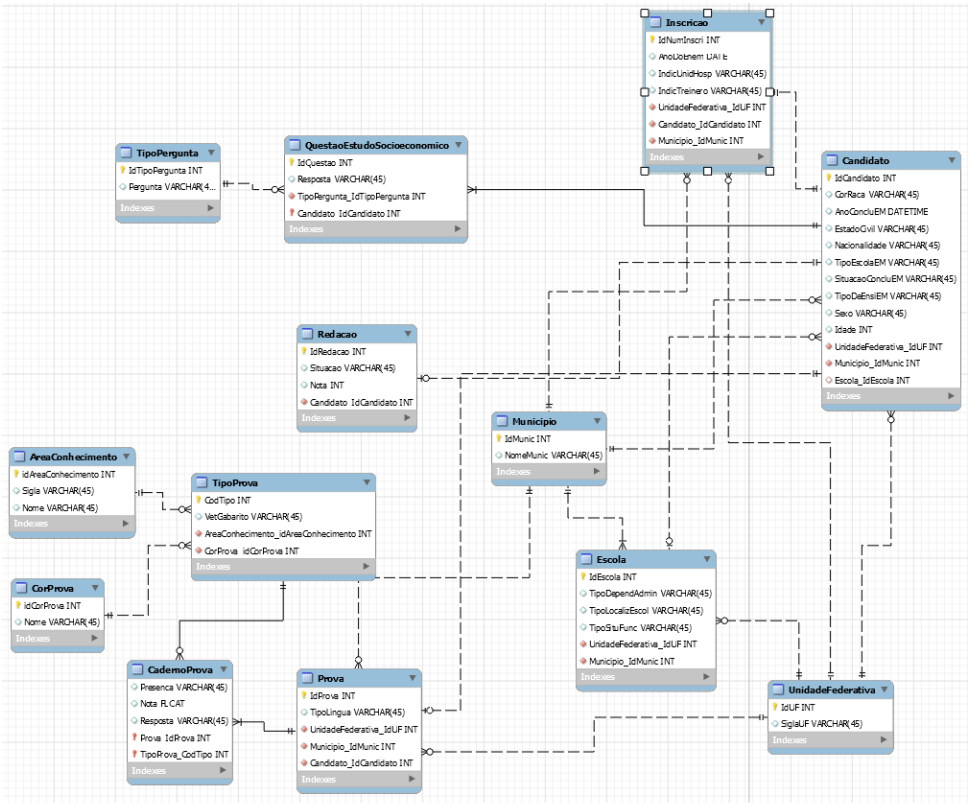

Figura 2: Modelo Relacional Enem 2015

3 Normalizando o Banco

A normalização é utilizada para validar um projeto relacional. Existem três formas normais. Se um projeto relacional não está nem na primeira forma normal, sua implementação é obsoleta. Caso o Banco não esteja na terceira forma normal, existirão problemas de inserção, remoção e atualização.

3.1 Primeira forma normal

A primeira forma normal diz que o valor de uma coluna de uma tabela é indivisível.

3.2 Segunda forma normal

Uma tabela está na segunda forma normal se ela está na primeira forma normal e todo atributo do complemento de uma chave candidata é totalmente funcionalmente dependente daquela chave.

3.3 Terceira forma normal

Uma relação está na terceira forma normal se, e somente se, estiver na segunda forma normal e todos os atributos não-chave forem dependentes não-transitivos da chave primária.

NOME DA VARIÁVEL	Descrição	Variáveis Categóricas		Tamanho	Tipo
		Categoria	Descrição		
DADOS DE INSCRIÇÃO					
NU_INSCRICAO	Número de inscrição[1]			12	Numérica
NU_ANO	Ano do Enem			4	Numérica
CO_MUNICIPIO_RESIDENCIA	Código do município de residência			7	Numérica
	1º dígito: Região				
	1º e 2º dígitos: UF				
	3º, 4º, 5º e 6º dígitos: Município				
	7º dígito: dígito verificador				
NO_MUNICIPIO_RESIDENCIA	Nome do município de residência			150	Alfanumérica
CO_UF_RESIDENCIA	Código da Unidade da Federação de residência			2	Numérica
SG_UF_RESIDENCIA	Sigla da Unidade da Federação de residência			2	Alfanumérica
IN_ESTUDA_CLASSE_HOSPITALAR	Indicador de inscrição em Unidade Hospitalar	1	Sim	1	Numérica
		0	Não		
IN_TREINEIRO	Indica se o inscrito fez a prova com intuito de apenas treinar seus conhecimentos[2]	1	Sim	1	Numérica
		0	Não		

Inscrição: NU_INSCRICAO, NU_ANO, CO_MUNICIPIO_RESIDENCIA, NO_MUNICIPIO_RESIDENCIA, CO_UF_RESIDENCIA, SG_UF_RESIDENCIA, IN_ESTUDA_CLASSE_HOSPITALAR, IN_TREINEIRO

Primeira forma normal: **Não há atributos aninhados**

Segunda forma normal: **NU_INSCRICAO - {NU_ANO, IN_ESTUDA_CLASSE_HOSPITALAR, IN_TREINEIRO, CO_MUNICIPIO_RESIDENCIA, NO_MUNICIPIO_RESIDENCIA, CO_UF_RESIDENCIA, SG_UF_RESIDENCIA}**

Terceira forma normal: **NU_INSCRICAO - {NU_ANO, IN_ESTUDA_CLASSE_HOSPITALAR, IN_TREINEIRO, CO_MUNICIPIO_RESIDENCIA, NO_MUNICIPIO_RESIDENCIA, CO_UF_RESIDENCIA, SG_UF_RESIDENCIA}**

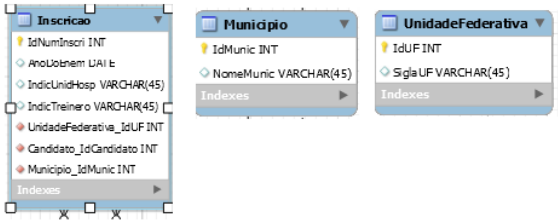

Figura 3: Normalizacao Enem 2015

4 Utilizando o Connector/Python3

Em ciência da computação, um cursor de banco de dados é uma estrutura de controle que permite percorrer sobre os registros em um banco de dados. Os cursores facilitam o processamento subsequente em conjunto com o percorrimento, tal como recuperação, adição e remoção de registros de banco de dados. A característica de percorrimento do cursor de banco de dados faz os cursores semelhantes ao conceito iterador de linguagens de programação.
Cursores são usados pelos programadores de banco de dados para processar linhas individuais

retornadas pelas consultas do sistema de banco de dados. Cursores permitem a manipulação de conjuntos de resultados completos de uma vez. Neste cenário, um cursor permite que linhas em um conjunto de resultados sejam processadas sequencialmente. Dessa forma surge a ferramenta Connector que nos permite atualizar, adicionar e remover registros do banco de dados. Primeiro é necessário estabilizar uma conexão com o banco desejado, em seguida, é chamado uma função para inicializar o cursor. O cursor fica em estado de espera até que se necessite realizar uma query.

4.1 Processo de Extract, Transform and Load

Extract, Transform, Load ou ETL ou Extração, Transformação, Carregamento, são ferramentas de software cuja função é a extração de dados de uma determinada fonte. A extração e carregamento são obrigatórios para o processo de inserção no banco de dados, sendo a transformação/limpeza as vezes opcional, porém, recomendada. O processo utilizado neste trabalho se deu com o auxílio da ferramenta connector onde com no cursor aberto foi procurado as informações do .csv fornecido pelo INEP e extraídas para um novo arquivo onde já foi transformado no formato apropriado para a inserção dos dados no MySQL.

4.2 Realizado Selects, Criando Views, Triggers e Procedures

Com o banco implementado e após a construção do programa com o auxílio da biblioteca Connector do Python, ao acessar o programa você pode realizar operações de inserção, remoção e atualização bem como de seleção e criação de Views, Triggers e Procedures.

4.3 Selects

O Select é o comando em SQL utilizado para seleções de dados de uma ou mais tabelas. Nosso programa abre um cursor onde o usuário pode digitar o que deseja visualizar no banco e o cursor retorna, caso seja validado, a seleção para o usuário.

4.4 Views

As visões ou Views, são tabelas virtuais criada a partir de um Select. Assim, podemos criar tabelas para visualização de dados mais rapidamente do que ter que escrever a querie inteira do select e acessa-la somente com um SELECT * FROM View. Criamos varias views em nosso programa, para ver as notas finais dos alunos, para ver os candidatos de um dterminado municipio e para ver a média das notas por sexo.

4.5 Triggers

Um Trigger ou gatilho é um objeto de banco de dados, associado a uma tabela, definido para ser disparado, respondendo a um evento em particular. A trigger implementada no programa realiza um backup e salva a tabela antes dela ser atualizada.

4.6 Procedures

Uma procedure ou processo é, como no trigger, a utilização de comandos da própria língugagem, neste caso, para a construção de uma rotina. Isto é, a construção de um programa que executa um conjunto de instruções que serão executadas no próprio banco de dados. Assim, a procedure implementada a seguir gera uma view onde pode-se escolher quantos alunos listar para a melhor nota de uma determinada escola ou unidade federal ou município. Exemplo, x melhores alunos de y, onde x é o número de alunos e y o local.

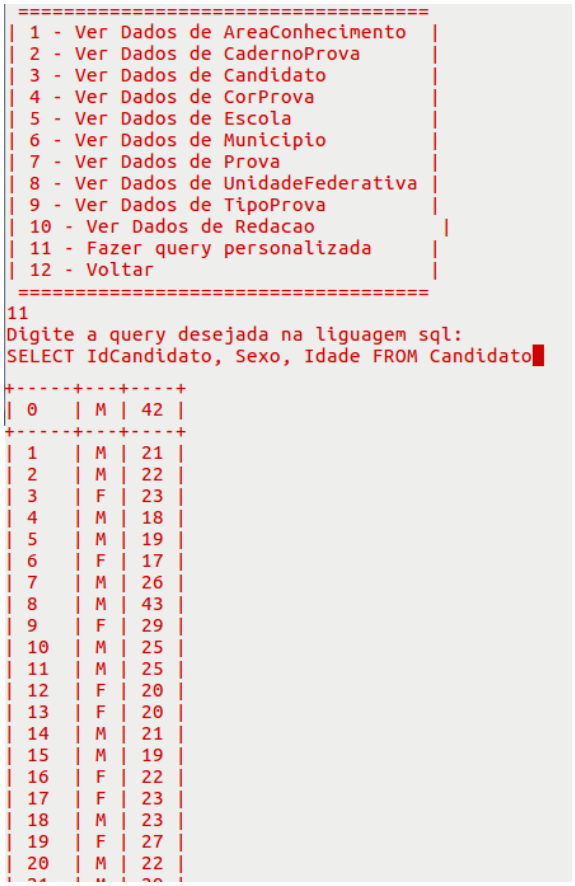

Figura 4: Selection Enem 2015

```
CREATE VIEW NotaFinalAluno AS
        SELECT
                IdCandidato, (SUM(CadernoProva.Nota)+Redacao.Nota)/5 as
Nota
        FROM
                Candidato
        JOIN
                UnidadeFederativa ON UnidadeFederativa.idUF =
Candidato.UnidadeFederativa_idUF
        JOIN
                Prova ON Prova.Candidato_idCandidato =
Candidato.idCandidato
        JOIN
                CadernoProva ON CadernoProva.Prova_idProva =
Prova.idProva
        JOIN
                Redacao ON Redacao.Candidato_IdCandidato = IdCandidato
        GROUP BY IdCandidato;
CREATE VIEW SexoIdade AS SELECT Sexo, Idade FROM Candidato;

CREATE VIEW CandidatosDF AS SELECT Candidato.* FROM Candidato JOIN
UnidadeFederativa ON UnidadeFederativa.idUF =
Candidato.UnidadeFederativa_idUF WHERE SiglaUF = 'DF';

CREATE VIEW CandidatosSP AS SELECT Candidato.* FROM Candidato JOIN
UnidadeFederativa ON UnidadeFederativa.idUF =
Candidato.UnidadeFederativa_idUF WHERE SiglaUF = 'SP';

CREATE VIEW CandidatosRJ AS SELECT Candidato.* FROM Candidato JOIN
UnidadeFederativa ON UnidadeFederativa.idUF =
Candidato.UnidadeFederativa_idUF WHERE SiglaUF = 'RJ';

CREATE VIEW MediaHomens AS SELECT AVG(Nota) AS Media FROM Candidato
NATURAL JOIN NotaFinalAluno WHERE Sexo = 'M' GROUP BY Sexo;

CREATE VIEW MediaMulheres AS SELECT AVG(Nota) AS Media FROM Candidato
NATURAL JOIN NotaFinalAluno WHERE Sexo = 'F' GROUP BY Sexo;
```

Figura 5: View Enem 2015

```
DROP TRIGGER IF EXISTS backup_candidato;
CREATE TABLE IF NOT EXISTS backup_candidato SELECT * FROM Candidato
LIMIT 0;
CREATE TRIGGER backup_candidato BEFORE UPDATE ON Candidato FOR EACH ROW
  INSERT INTO backup_candidato
    (idCandidato, CorRaca, AnoConcluEM, EstadoCivil, Nacionalidade,
TipoEscolaEM,SituacaoConcluEM,TipoDeEnsiEM,Sexo,Idade,UnidadeFederativa_I
Municipio_IdMunic, Escola_IdEscola)
  VALUES
    (OLD.idCandidato, OLD.CorRaca, OLD.AnoConcluEM, OLD.EstadoCivil,
OLD.Nacionalidade, OLD.TipoEscolaEM, OLD.SituacaoConcluEM,
OLD.TipoDeEnsiEM, OLD.Sexo, OLD.Idade, OLD.UnidadeFederativa_IdUF,
OLD.Municipio_IdMunic, OLD.Escola_IdEscola);

DROP TRIGGER IF EXISTS backup_caderno_prova;
CREATE TABLE IF NOT EXISTS backup_caderno_prova SELECT * FROM
CadernoProva LIMIT 0;
CREATE TRIGGER backup_caderno_prova BEFORE UPDATE ON CadernoProva FOR
EACH ROW
  INSERT INTO backup_caderno_prova
    (Prova_IdProva, Presenca, Nota, Resposta, TipoProva_CodTipo)
  VALUES
    (OLD.Prova_IdProva, OLD.Presenca, OLD.Nota, OLD.Resposta,
OLD.TipoProva_CodTipo);


DROP TRIGGER IF EXISTS backup_escola;
CREATE TABLE IF NOT EXISTS backup_escola SELECT * FROM Escola LIMIT 0;
CREATE TRIGGER backup_escola BEFORE UPDATE ON Escola FOR EACH ROW
  INSERT INTO backup_escola
    (IdEscola, TipoDependAdmin, TipoLocalizEscol, TipoSituFunc,
UnidadeFederativa_IdUF, Municipio_IdMunic)
  VALUES
    (OLD.IdEscola, OLD.TipoDependAdmin, OLD.TipoLocalizEscol,
OLD.TipoSituFunc, OLD.UnidadeFederativa_IdUF, OLD.Municipio_IdMunic);

DROP TRIGGER IF EXISTS backup_prova;
CREATE TABLE IF NOT EXISTS backup_prova SELECT * FROM Prova LIMIT 0;
CREATE TRIGGER backup_prova BEFORE UPDATE ON Prova FOR EACH ROW
  INSERT INTO backup_prova
    (IdProva, TipoLingua, Candidato_idCandidato, UnidadeFederativa_IdUF,
Municipio_IdMunic)
  VALUES
    (OLD.IdProva, OLD.TipoLingua, OLD.Candidato_idCandidato,
OLD.UnidadeFederativa_IdUF, OLD.Municipio_IdMunic);

DROP TRIGGER IF EXISTS backup_redacao;
CREATE TABLE IF NOT EXISTS backup_redacao SELECT * FROM Redacao LIMIT 0;
CREATE TRIGGER backup_redacao BEFORE UPDATE ON Redacao FOR EACH ROW
  INSERT INTO backup_redacao
    (IdRedacao, Situacao, Candidato_idCandidato, Nota)
  VALUES
    (OLD.IdRedacao, OLD.Situacao, OLD.Candidato_idCandidato, OLD.Nota);
```

Figura 6: Trigger Enem 2015

```
DROP PROCEDURE IF EXISTS melhores_alunos_estado;
DELIMITER //
CREATE PROCEDURE melhores_alunos_estado
(IN numero_candidatos INT, IN estado VARCHAR(45))
  BEGIN
  DROP TABLE IF EXISTS melhores_alunos_estado;
  CREATE TABLE melhores_alunos_estado
    SELECT
      Candidato.*, Nota
    FROM
                Candidato
        JOIN
                UnidadeFederativa ON UnidadeFederativa.idUF =
Candidato.UnidadeFederativa_idUF
        JOIN
                NotaFinalAluno ON NotaFinalAluno.IdCandidato =
Candidato.IdCandidato
        WHERE
      SiglaUF = estado
        ORDER BY Nota DESC LIMIT numero_candidatos;
        SELECT * FROM melhores_alunos_estado;
        END //


DROP PROCEDURE IF EXISTS melhores_alunos_escola;
DELIMITER //
CREATE PROCEDURE melhores_alunos_escola
(IN numero_candidatos INT, IN escola VARCHAR(45))
        BEGIN
        DROP TABLE IF EXISTS melhores_alunos_escola;
        CREATE TABLE melhores_alunos_escola
                SELECT
                  Candidato.*, Nota
                FROM
                        Candidato
                JOIN
                        UnidadeFederativa ON UnidadeFederativa.idUF =
Candidato.UnidadeFederativa_idUF
                JOIN
                        Prova ON Prova.Candidato_idCandidato =
Candidato.IdCandidato
                JOIN
                        CadernoProva ON CadernoProva.Prova_idProva =
Prova.idProva
                WHERE
                        CASE WHEN escola = 'publica' THEN
                                TipoEscolaEM = 2
                        WHEN escola = 'privada' THEN
                                TipoEscolaEM = 3
                        ELSE
                                TipoEscolaEM = -1
                        END
                ORDER BY Nota DESC LIMIT numero_candidatos;
        SELECT * FROM melhores_alunos_escola;
        END //
```

Figura 7: Procedure Enem 2015

5 Conclusão

Neste trabalho foram realizados diversas técnicas aprendidas ao longo do curso da matéria Banco de Dados, onde os integrantes do grupo puderam testar seus conhecimentos realizando a construção de um Banco tendo somente os microdados fornecidos pelo INEP. Foi necessário modelar os dados, aplicar formas normais e por fim deixar os dados organizados para uma aplicação ser capaz de utilizá-la como quiser.

6 Referencias

https://dev.mysql.com/doc/connector-python/en/

McKinney, Wes. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. "O'Reilly Media, Inc.", 2012. NBR 6023