IF2240 - Basis Data

Tugas Besar Milestone III
Pemodelan Entity-Relationship

Kelompok 4

Anggota:

Kelvin Rayhan Alkarim	**13521005**
Azmi Hasna Zahrani	**13521006**
Jason Rivalino	**13521008**
Muhhamad Syauqi Janattan	**13521014**
Varraz Hazzandra Abrar	**13521020**
Raditya Naufal Abiyu	**13521022**

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

2023

DAFTAR ISI

BAB I

RELATIONAL DATABASE DESIGN

I. Skema basis data awal sebelum normalisasi

Skema dari basis data awal yang sebelum dinormalisasi adalah sebagai berikut:

Tabel dan Atribut:

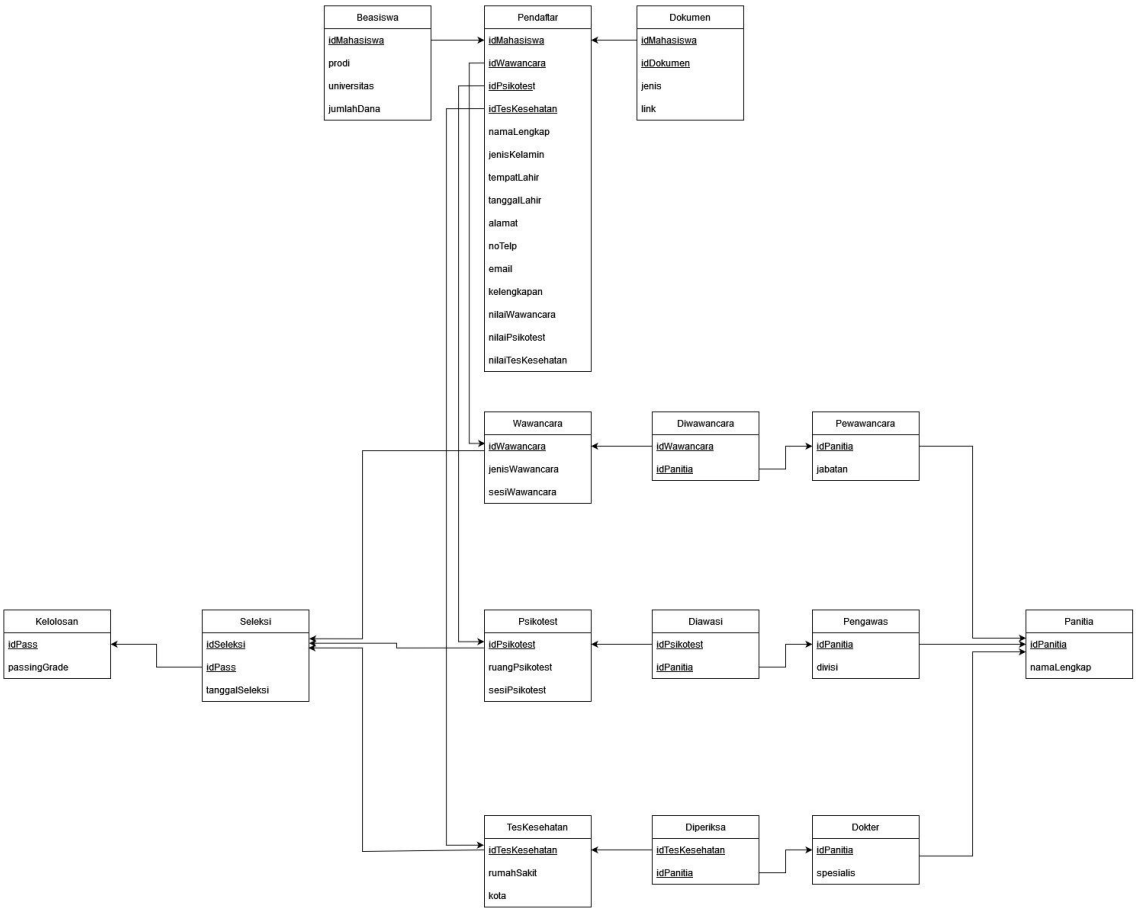
- Pendaftar: { idMahasiswa, idWawancara, idPsikotest, idTesKesehatan, namaLengkap, jenisKelamin, tempatLahir, tanggalLahir, alamat, noTelp, email, kelengkapan, nilaiWawancara, nilaiPsikotest, nilaiTesKesehatan }
- Dokumen: { idMahasiswa, idDokumen, jenis, link }
- Beasiswa: { idMahasiswa, prodi, universitas, jumlahDana }
- Seleksi: { idSeleksi, idPass, tanggalSeleksi }
- Kelolosan: { idPass, passingGrade }
- Panitia: { idPanitia, namaLengkap }
- Wawancara: { idWawancara, jenisWawancara, sesiWawancara }
- Diwawancara: { idWawancara, idPanitia }
- Pewawancara: { idPanitia, jabatan }
- Psikotest: { idPsikotest, ruangPsikotest, sesiPsikotest }
- Diawasi: { idPsikotest, idPanitia }
- Pengawas: { idPanitia, divisi }
- TesKesehatan: { idTesKesehatan, rumahSakit, kota }
- Diperiksa: { idTesKesehatan, idPanitia }
- Dokter: { idPanitia, spesialis }

Foreign Key:

- Dokumen (idMahasiswa) -> Pandaftar (idMahasiswa)
- Beasiswa (idMahasiswa) -> Pandaftar (idMahasiswa)
- Seleksi (idPass) -> Kelolosan (idPass)
- Pendaftar (idWawancara) -> Wawancara (idWawancara)
- Pendaftar (idPsikotest) -> Psikotest (idPsikotest)
- Pendaftar (idTesKesehatan) -> TesKesehatan (idTesKesehatan)
- Wawancara (idWawancara) -> Seleksi (idSeleksi)
- Psikotest (idPsikotest) -> Seleksi (idSeleksi)
- TesKesehatan (idTesKesehatan) -> Seleksi (idSeleksi)
- Diwawancara (idWawancara) -> Wawancara (idWawancara)
- Diawasi (idPsikotest) -> Psikotest (idPsikotest)

- Diperiksa (idTesKesehatan) -> TesKesehatan (idTesKesehatan)
- Diwawancara (idTesKesehatan) -> Pewawancara (idPanitia)
- Diawasi (idPanitia) -> Pengawas (idPanitia)
- Diperiksa (idPanitia) -> Dokter (idPanitia)
- Pewawancara (idPanitia) -> Panitia (idPanitia)
- Pengawas (idPanitia) -> Panitia (idPanitia)
- Dokter (idPanitia) -> Panitia (idPanitia)

Gambar dari skema basis data sebelum normalisasi adalah sebagai berikut:



II. Identifikasi *Functional Dependencies* (FD)

Bentuk dari *Functional Dependencies* yang ada berdasarkan skema basis data sebelumnya adalah sebagai berikut:

1. Relational Database Pendaftar

   Tabel dan Atribut dari relational database ini adalah sebagai berikut:

Pendaftar: { idMahasiswa, idWawancara, idPsikotest, idTesKesehatan, namaLengkap, jenisKelamin, tempatLahir, tanggalLahir, alamat, noTelp, email, kelengkapan, nilaiWawancara, nilaiPsikotest, nilaiTesKesehatan }

   Berdasarkan informasi dari tabel dan atribut ini, semua atribut yang berisikan informasi terkait mahasiswa (namaLengkap, jenisKelamin, tempatLahir, tanggalLahir, alamat, noTelp, email, kelengkapan), semuanya bergantung kepada idMahasiswa yang mendaftar. Selain itu, semua nilai yang didapatkan pada tes seleksi (wawancara, psikotes, dan tesKesehatan), nilai-nilai yang ada bergantung pada id dari seleksi yang diikutinya. Sehingga, berdasarkan hal ini, bentuk *Functional Depedencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

idMahasiswa -> namaLengkap, jenisKelamin, tempatLahir, tanggalLahir, alamat, noTelp, email, kelengkapan idWawancara → nilaiWawancara idPsikotest → nilaiPsikotest idTesKesehatan → nilaiTesKesehatan

2. Relational Database Dokumen

   Tabel dan Atribut dari relational database ini adalah sebagai berikut:

Dokumen: { idMahasiswa, idDokumen, jenis, link }

   Berdasarkan informasi dari tabel dan atribut ini, semua atribut yang berisikan informasi terkait dokumen (jenis dan link dokumen), semuanya bergantung kepada idMahasiswa yang mendaftar dan idDokumen yang

dilampirkan. Selain itu, idMahasiswa sendiri juga bergantung kepada idDokumen karena setiap mahasiswa melampirkan dokumen dengan jenis dan jumlah yang berbeda-beda. Sehingga, berdasarkan hal ini, bentuk *Functional Depedencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

```
idDokumen -> idMahasiswa
idMahasiswa, idDokumen -> jenis, link
```

3. Relational Database Beasiswa

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

```
Beasiswa: { idMahasiswa, prodi, universitas, jumlahDana }
```

Berdasarkan informasi dari tabel dan atribut ini, semua atribut yang berisikan informasi terkait mahasiswa (prodi, universitas, jumlahDana), semuanya bergantung kepada idMahasiswa yang mendaftar. Sehingga, berdasarkan hal ini, bentuk *Functional Dependencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

```
idMahasiswa -> universitas, prodi, jumlahDana
```

4. Relational Database Seleksi

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

```
Seleksi: { idSeleksi, idPass, tanggalSeleksi }
```

Berdasarkan informasi dari tabel dan atribut ini, semua atribut yang berisikan informasi terkait seleksi yaitu tanggalSeleksi, atribut ini bergantung kepada idSeleksi dan idPass dari tes yang diselenggarakan. Selain itu, idPass sendiri juga bergantung dari idSeleksi karena batas nilai passing grade bergantung

pada seleksi yang diikuti oleh pendaftar. Sehingga, berdasarkan hal ini, bentuk *Functional Depedencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

```
idSeleksi -> idPass
idSeleksi, idPass -> tanggalSeleksi
```

5. Relational Database Kelolosan

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

```
Kelolosan: { idPass, passingGrade }
```

Berdasarkan informasi dari tabel dan atribut ini, nilai passing grade dari sebuah tes bergantung kepada idPass dari tes yang diselenggarakan. Sehingga, berdasarkan hal ini, bentuk *Functional Depedencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

```
idPass -> passingGrade
```

6. Relational Database Panitia

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

```
Panitia: { idPanitia, namaLengkap }
```

Berdasarkan informasi dari tabel dan atribut ini, informasi terkait panitia yaitu namaLengkap akan bergantung kepada idPanitia yang dimiliki. Sehingga, berdasarkan hal ini, bentuk *Functional Depedencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

idPanitia -> namaLengkap

7. Relational Database Wawancara

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

Wawancara: { idWawancara, jenisWawancara, sesiWawancara }

Berdasarkan informasi dari tabel dan atribut ini, informasi terkait seleksi wawancara yaitu jenisWawancara dan sesiWawancara akan bergantung kepada idWawancara yang dimiliki. Sehingga, berdasarkan hal ini, bentuk *Functional Depedencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

idWawancara -> jenisWawancara, sesiWawancara

8. Relational Database Diwawancara

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

Diwawancara: { idWawancara, idPanitia }

Berdasarkan informasi dari tabel dan atribut ini, informasi yang terdapat dalam relational ini tidak dapat dicari bentuk functional dependenciesnya, karena untuk satu wawancara bisa diwawancarai oleh banyak panitia, begitupun sebaliknya untuk satu panitia bisa mewawancarai banyak pendaftar di sesi wawancara yang berbeda. Sehingga, tidak memenuhi syarat dari terbentuknya functional dependencies.

9. Relational Database Pewawancara

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

Pewawancara: { idPanitia, jabatan }

Berdasarkan informasi dari tabel dan atribut ini, informasi terkait pewawancara yaitu jabatan akan akan bergantung kepada idPanitia yang dimiliki. Sehingga, berdasarkan hal ini, bentuk *Functional Depedencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

idPanitia -> jabatan

10. Relational Database Psikotest

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

Psikotest: { idPsikotest, ruangPsikotest, sesiPsikotest }

Berdasarkan informasi dari tabel dan atribut ini, informasi terkait seleksi psikotest yaitu ruangPsikotest dan sesiPsikotest akan bergantung kepada idPsikotest yang dimiliki. Sehingga, berdasarkan hal ini, bentuk *Functional Depedencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

idPsikotest -> ruangPsikotest, sesiPsikotest

11. Relational Database Diawasi

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

Diawasi: { idPsikotest, idPanitia }

Berdasarkan informasi dari tabel dan atribut ini, informasi yang terdapat dalam relational ini tidak dapat dicari bentuk functional dependenciesnya, karena untuk satu psikotest bisa diawasi oleh banyak panitia, begitupun sebaliknya untuk

satu panitia bisa mengawasi banyak psikotest di ruangan yang berbeda. Sehingga, tidak memenuhi syarat dari terbentuknya functional dependencies.

12. Relational Database Pengawas

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

Pengawas: { idPanitia, divisi }

Berdasarkan informasi dari tabel dan atribut ini, informasi terkait pengawas yaitu divisi akan akan bergantung kepada idPanitia yang dimiliki. Sehingga, berdasarkan hal ini, bentuk *Functional Depedencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

idPanitia -> divisi

13. Relational Database TesKesehatan

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

TesKesehatan: { idTesKesehatan, rumahSakit, kota }

Berdasarkan informasi dari tabel dan atribut ini, informasi terkait seleksi tes kesehatan yaitu rumahSakit dan kota akan bergantung kepada idTesKesehatan yang dimiliki. Sehingga, berdasarkan hal ini, bentuk *Functional Depedencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

idTesKesehatan -> rumahSakit, kota

14. Relational Database Diperiksa

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

Diperiksa: { idTesKesehatan, idPanitia }

Berdasarkan informasi dari tabel dan atribut ini, informasi yang terdapat dalam relational ini tidak dapat dicari bentuk functional dependenciesnya, karena untuk satu tes kesehatan bisa diperiksa oleh banyak dokter, begitupun sebaliknya untuk satu dokter bisa mengawasi banyak tes kesehatan di rumah sakit yang berbeda. Sehingga, tidak memenuhi syarat dari terbentuknya functional dependencies.

15. Relational Database Dokter

Tabel dan Atribut dari relational database ini adalah sebagai berikut:

Dokter: { idPanitia, spesialis }

Berdasarkan informasi dari tabel dan atribut ini, informasi terkait dokter yaitu spesialis akan akan bergantung kepada idPanitia yang dimiliki. Sehingga, berdasarkan hal ini, bentuk *Functional Depedencies* yang didapat dari relational database Pendaftar adalah sebagai berikut:

idPanitia -> spesialis

III. Identifikasi Bentuk Normal Relasi

1. Pendaftar

idMahasiswa -> namaLengkap, jenisKelamin, tempatLahir, tanggalLahir, alamat, noTelp, email, kelengkapan
idWawancara → nilaiWawancara
idPsikotest → nilaiPsikotest
idTesKesehatan → nilaiTesKesehatan

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada sudah dalam bentuk *canonical cover* dengan candidate keys yang ada antara lain idMahasiswa, idWawancara, idPsikotest, dan idTesKesehatan. Namun bentuk ini masih hanya dalam bentuk 1NF dengan alasan karena masih terdapat partial dependencies dalam functional dependencies yang ada. Sehingga masih perlu dilakukan normalisasi terlebih dahulu.

2. Dokumen

idDokumen -> idMahasiswa
idMahasiswa, idDokumen -> jenis, link

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada masih belum dalam bentuk canonical cover sehingga perlu disederhanakan lagi menjadi bentuk seperti berikut:

idDokumen -> idMahasiswa, jenis, link

Dari bentuk ini, candidate keys yang didapatkan hanya idDokumen saja. Karena idDokumen sudah langsung mengarah pada semua atribut lainnya, maka tidak ada bentuk partial ataupun transitive dependencies sehingga bentuk tabel ini sudah dalam kondisi BCNF.

3. Beasiswa

```
idMahasiswa -> universitas, prodi, jumlahDana
```

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada sudah dalam bentuk *canonical cover* dengan candidate keys yang ada yaitu idMahasiswa. Karena idMahasiswa sudah langsung mengarah pada semua atribut lainnya, maka tidak ada bentuk partial ataupun transitive dependencies sehingga bentuk tabel ini sudah dalam kondisi BCNF.

4. Seleksi

```
idSeleksi -> idPass
idSeleksi, idPass -> tanggalSeleksi
```

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada masih belum dalam bentuk canonical cover sehingga perlu disederhanakan lagi menjadi bentuk seperti berikut:

```
idDokumen -> idMahasiswa, jenis, link
```

Dari bentuk ini, candidate keys yang didapatkan hanya idDokumen saja. Karena idDokumen sudah langsung mengarah pada semua atribut lainnya, maka tidak ada bentuk partial ataupun transitive dependencies sehingga bentuk tabel ini sudah dalam kondisi BCNF.

5. Kelolosan

```
idPass -> passingGrade
```

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada sudah dalam bentuk *canonical cover* dengan candidate keys yang ada yaitu idPass. Karena idPass sudah langsung mengarah pada semua atribut lainnya, maka tidak ada bentuk partial ataupun transitive dependencies sehingga bentuk tabel ini sudah dalam kondisi BCNF.

6. Kelolosan

idPanitia -> namaLengkap

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada sudah dalam bentuk *canonical cover* dengan candidate keys yang ada yaitu idPanitia. Karena idPanitia sudah langsung mengarah pada semua atribut lainnya, maka tidak ada bentuk partial ataupun transitive dependencies sehingga bentuk tabel ini sudah dalam kondisi BCNF.

7. Wawancara

idWawancara -> jenisWawancara, sesiWawancara

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada sudah dalam bentuk *canonical cover* dengan candidate keys yang ada yaitu idWawancara. Karena idWawancara sudah langsung mengarah pada semua atribut lainnya, maka tidak ada bentuk partial ataupun transitive dependencies sehingga bentuk tabel ini sudah dalam kondisi BCNF.

8. Pewawancara

idPanitia -> jabatan

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada sudah dalam bentuk *canonical cover* dengan candidate keys yang ada yaitu idPanitia. Karena idPanitia sudah langsung mengarah pada semua atribut lainnya, maka tidak ada

bentuk partial ataupun transitive dependencies sehingga bentuk tabel ini sudah dalam kondisi BCNF.

9. Psikotest

```
idPsikotest -> ruangPsikotest, sesiPsikotest
```

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada sudah dalam bentuk *canonical cover* dengan candidate keys yang ada yaitu idPsikotest. Karena idPsikotest sudah langsung mengarah pada semua atribut lainnya, maka tidak ada bentuk partial ataupun transitive dependencies sehingga bentuk tabel ini sudah dalam kondisi BCNF.

10. Pengawas

```
idPanitia -> divisi
```

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada sudah dalam bentuk *canonical cover* dengan candidate keys yang ada yaitu idPanitia. Karena idPanitia sudah langsung mengarah pada semua atribut lainnya, maka tidak ada bentuk partial ataupun transitive dependencies sehingga bentuk tabel ini sudah dalam kondisi BCNF.

11. TesKesehatan

```
idTesKesehatan -> rumahSakit, kota
```

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada sudah dalam bentuk *canonical cover* dengan candidate keys yang ada yaitu idTesKesehatan. Karena idTesKesehatan sudah langsung mengarah pada semua atribut lainnya, maka tidak ada bentuk partial ataupun transitive dependencies sehingga bentuk tabel ini sudah dalam kondisi BCNF.

12. Dokter

idPanitia -> spesialis

Berdasarkan bentuk *functional dependencies* ini, bentuk yang ada sudah dalam bentuk *canonical cover* dengan candidate keys yang ada yaitu idPanitia. Karena idPanitia sudah langsung mengarah pada semua atribut lainnya, maka tidak ada bentuk partial ataupun transitive dependencies sehingga bentuk tabel ini sudah dalam kondisi BCNF.

IV. Langkah-langkah Normalisasi Skema Basis Data

Berdasarkan identifikasi bentuk normal sebelumnya, hanya satu tabel yang perlu dilakukan normalisasi yaitu tabel pendaftar dengan *functional depedencies* sebagai berikut:

idMahasiswa -> namaLengkap, jenisKelamin, tempatLahir, tanggalLahir, alamat, noTelp, email, kelengkapan idWawancara → nilaiWawancara idPsikotest → nilaiPsikotest idTesKesehatan → nilaiTesKesehatan

Dari bentuk ini, proses normalisasi yang perlu dilakukan adalah dengan memecah menjadi beberapa tabel agar tidak mengalami bentuk *partial depedency*. Setelah proses pengubahan, akan terbentuk beberapa tabel sebagai berikut:

infoPendaftar: { idMahasiswa, namaLengkap, jenisKelamin, tempatLahir, tanggalLahir, alamat, noTelp, email, kelengkapan }
nilaiWawancaraPendaftar: { idWawancara, nilaiWawancara }
nilaiPsikotestPendaftar: { idPsikotest, nilaiPsikotest }
nilaiTesKesehatanPendaftar: { idTesKesehatan, nilaiTesKesehatan }
tesPendaftar: { idMahasiswa, idWawancara, idPsikotest, idTesKesehatan }

Setelah terbentuk tabel ini, dilakukan pengecekan kondisi normal terhadap berbagai tabel yang terbentuk. Karena untuk semua tabel yang terbentuk, *primary key* yang ada sudah mengarah ke semua atributnya, maka kondisi tabel ini sudah dalam bentuk BCNF.

V. Skema Basis Data Hasil Normalisasi

Berdasarkan hasil normalisasi yang dilakukan sebelumnya, bentuk skema basis data baru yang terbentuk adalah sebagai berikut:

Tabel dan Atribut:

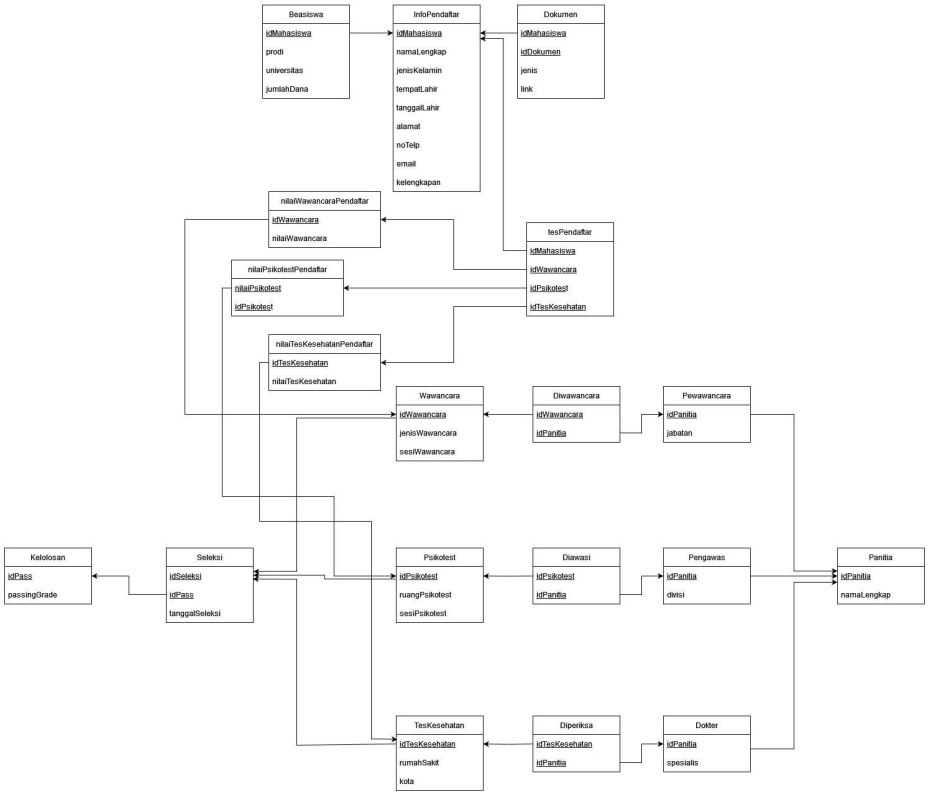
- infoPendaftar: { idMahasiswa, namaLengkap, jenisKelamin, tempatLahir, tanggalLahir, alamat, noTelp, email, kelengkapan }
- nilaiWawancaraPendaftar: { idWawancara, nilaiWawancara }
- nilaiPsikotestPendaftar: { idPsikotest, nilaiPsikotest }
- nilaiTesKesehatanPendaftar: { idTesKesehatan, nilaiTesKesehatan }
- tesPendaftar: { idMahasiswa, idWawancara, idPsikotest, idTesKesehatan }
- Dokumen: { idMahasiswa, idDokumen, jenis, link }
- Beasiswa: { idMahasiswa, prodi, universitas, jumlahDana }
- Seleksi: { idSeleksi, idPass, tanggalSeleksi }
- Kelolosan: { idPass, passingGrade }
- Panitia: { idPanitia, namaLengkap }
- Wawancara: { idWawancara, jenisWawancara, sesiWawancara }
- Diwawancara: { idWawancara, idPanitia }
- Pewawancara: { idPanitia, jabatan }
- Psikotest: { idPsikotest, ruangPsikotest, sesiPsikotest }
- Diawasi: { idPsikotest, idPanitia }
- Pengawas: { idPanitia, divisi }
- TesKesehatan: { idTesKesehatan, rumahSakit, kota }
- Diperiksa: { idTesKesehatan, idPanitia }
- Dokter: { idPanitia, spesialis }

Foreign Key:

- Dokumen (idMahasiswa) -> infoPandaftar (idMahasiswa)
- Beasiswa (idMahasiswa) -> infoPandaftar (idMahasiswa)
- tesPendaftar (idMahasiswa) -> infoPandaftar (idMahasiswa)
- tesPendaftar (idWawancara) -> nilaiWawancaraPendaftar (idWawancara)
- nilaiWawancaraPendaftar (idWawancara) -> wawancara (idWawancara)

- tesPendaftar (idPsikotest) -> nilaiPsikotestPendaftar (idPsikotest)
- nilaiPsikotestPendaftar (idPsikotest) -> Psikotest (iidPsikotest)
- tesPendaftar (idTesKesehatan) -> nilaiTesKesehatanPendaftar (idTesKesehatan)
- nilaiTesKesehatanPendaftar (idTesKesehatan) -> TesKesehatan (idTesKesehatan)
- Seleksi (idPass) -> Kelolosan (idPass)
- Pendaftar (idWawancara) -> Wawancara (idWawancara)
- Pendaftar (idPsikotest) -> Psikotest (idPsikotest)
- Pendaftar (idTesKesehatan) -> TesKesehatan (idTesKesehatan)
- Wawancara (idWawancara) -> Seleksi (idSeleksi)
- Psikotest (idPsikotest) -> Seleksi (idSeleksi)
- TesKesehatan (idTesKesehatan) -> Seleksi (idSeleksi)
- Diwawancara (idWawancara) -> Wawancara (idWawancara)
- Diawasi (idPsikotest) -> Psikotest (idPsikotest)
- Diperiksa (idTesKesehatan) -> TesKesehatan (idTesKesehatan)
- Diwawancara (idTesKesehatan) -> Pewawancara (idPanitia)
- Diawasi (idPanitia) -> Pengawas (idPanitia)
- Diperiksa (idPanitia) -> Dokter (idPanitia)
- Pewawancara (idPanitia) -> Panitia (idPanitia)
- Pengawas (idPanitia) -> Panitia (idPanitia)
- Dokter (idPanitia) -> Panitia (idPanitia)

Gambar dari skema basis data setelah normalisasi adalah sebagai berikut:



BAB II

BUSINESS RULE

I. *Query-query* penerapan *business rule* pada basis data.

1. View untuk menampilkan jumlah pendaftar yang lulus, nilai rata-rata, nilai maksimal, dan nilai minimal pada setiap tahapan seleksi

a. Menampilkan jumlah pendaftar yang lulus pada seleksi administrasi

- Query pembuatan view:

```
CREATE VIEW dataSeleksiAdministrasi as
SELECT
COUNT(idMahasiswa) as jumlahLulusAdministrasi
FROM infoPendaftar
WHERE kelengkapan = 1;
```

```
MariaDB [lpdp2]> CREATE VIEW dataSeleksiAdministrasi as
    -> SELECT
    -> COUNT(idMahasiswa) as jumlahLulusAdministrasi
    -> FROM infoPendaftar
    -> WHERE kelengkapan = 1;
Query OK, 0 rows affected (0.006 sec)
```

- Query untuk menampilkan informasi

```
SELECT * from dataSeleksiAdministrasi;
```

```
MariaDB [lpdp2]> SELECT * from dataSeleksiAdministrasi;
+------------------------+
| jumlahLulusAdministrasi |
+------------------------+
|                     60 |
+------------------------+
1 row in set (0.006 sec)
```

b. Menampilkan jumlah pendaftar yang lulus pada seleksi wawancara

- Query pembuatan view:

```
CREATE VIEW dataSeleksiWawancara AS
        SELECT
        COUNT(CASE WHEN nwp.idWawancara = w.idWawancara
                AND w.idWawancara = s.idSeleksi
                AND s.idPass = l.idPass
                AND nwp.nilaiWawancara > l.passingGrade
                THEN 1
                ELSE NULL
                END) as jumlahLulusWawancara,
AVG(nwp.nilaiWawancara) as ratarata,
MAX(nwp.nilaiWawancara) as nilaiMaksimum,
MIN(nwp.nilaiWawancara) as nilaiMinimum
FROM nilaiWawancaraPendaftar as nwp, wawancara as w, seleksi as s, kelolosan
as l;
```

```
MariaDB [lpdp2]> CREATE VIEW dataSeleksiWawancara AS
    ->  SELECT
    ->  COUNT(CASE WHEN nwp.idWawancara = w.idWawancara
    ->          AND w.idWawancara = s.idSeleksi
    ->          AND s.idPass = l.idPass
    ->          AND nwp.nilaiWawancara > l.passingGrade
    ->          THEN 1
    ->          ELSE NULL
    ->          END) as jumlahLulusWawancara,
    -> AVG(nwp.nilaiWawancara) as ratarata,
    -> MAX(nwp.nilaiWawancara) as nilaiMaksimum,
    -> MIN(nwp.nilaiWawancara) as nilaiMinimum
    -> FROM nilaiWawancaraPendaftar as nwp, wawancara as w, seleksi as s, kelolosan as l;
Query OK, 0 rows affected (0.007 sec)
```

- Query untuk menampilkan informasi

```
SELECT * from dataSeleksiWawancara;
```

```
MariaDB [lpdp2]> SELECT * from dataSeleksiWawancara;
+---------------------+------------+---------------+--------------+
| jumlahLulusWawancara | ratarata   | nilaiMaksimum | nilaiMinimum |
+---------------------+------------+---------------+--------------+
|                  40 | 195.349667 |        396.28 |         3.58 |
+---------------------+------------+---------------+--------------+
1 row in set (0.261 sec)
```

c. Menampilkan jumlah pendaftar yang lulus pada seleksi psikotest

- Query pembuatan view:

```
CREATE VIEW dataSeleksiPsikotest AS
        SELECT
        COUNT(CASE WHEN npp.idPsikotest = p.idPsikotest
                AND p.idPsikotest = s.idSeleksi
                AND s.idPass = l.idPass
                AND npp.nilaiPsikotest > l.passingGrade
                THEN 1
                ELSE NULL
                END) as jumlahLulusPsikotest,
AVG(npp.nilaiPsikotest) as ratarata,
MAX(npp.nilaiPsikotest) as nilaiMaksimum,
MIN(npp.nilaiPsikotest) as nilaiMinimum
FROM nilaiPsikotestPendaftar as npp, psikotest as p, seleksi as s, kelolosan as l;
```

```
MariaDB [lpdp2]> CREATE VIEW dataSeleksiPsikotest AS
    ->  SELECT
    ->  COUNT(CASE WHEN npp.idPsikotest = p.idPsikotest
    ->          AND p.idPsikotest = s.idSeleksi
    ->          AND s.idPass = l.idPass
    ->          AND npp.nilaiPsikotest > l.passingGrade
    ->          THEN 1
    ->          ELSE NULL
    ->          END) as jumlahLulusPsikotest,
    -> AVG(npp.nilaiPsikotest) as ratarata,
    -> MAX(npp.nilaiPsikotest) as nilaiMaksimum,
    -> MIN(npp.nilaiPsikotest) as nilaiMinimum
    -> FROM nilaiPsikotestPendaftar as npp, psikotest as p, seleksi as s, kelolosan as l;
Query OK, 0 rows affected (0.006 sec)
```

- Query untuk menampilkan informasi

```
SELECT * from dataSeleksiPsikotest;
```

```
MariaDB [lpdp2]> SELECT * from dataSeleksiPsikotest;
+---------------------+------------+---------------+--------------+
| jumlahLulusPsikotest | ratarata   | nilaiMaksimum | nilaiMinimum |
+---------------------+------------+---------------+--------------+
|                  20 | 185.089000 |        383.76 |         3.98 |
+---------------------+------------+---------------+--------------+
1 row in set (0.019 sec)
```

d. Menampilkan jumlah pendaftar yang lulus pada seleksi tes kesehatan

- Query pembuatan view:

```
CREATE VIEW dataSeleksiTesKesehatan AS
        SELECT
        COUNT(CASE WHEN ntkp.idTesKesehatan = tk.idTesKesehatan
                AND tk.idTesKesehatan = s.idSeleksi
                AND s.idPass = l.idPass
                AND ntkp.nilaiTesKesehatan > l.passingGrade
                THEN 1
                ELSE NULL
                END) as jumlahTesKesehatan,
AVG(ntkp.nilaiTesKesehatan) as ratarata,
MAX(ntkp.nilaiTesKesehatan) as nilaiMaksimum,
MIN(ntkp.nilaiTesKesehatan) as nilaiMinimum
FROM nilaiTesKesehatanPendaftar as ntkp, tesKesehatan as tk, seleksi as s,
kelolosan as l;
```

```
MariaDB [lpdp2]> CREATE VIEW dataSeleksiTesKesehatan AS
    ->  SELECT
    ->  COUNT(CASE WHEN ntkp.idTesKesehatan = tk.idTesKesehatan
    ->          AND tk.idTesKesehatan = s.idSeleksi
    ->          AND s.idPass = l.idPass
    ->          AND ntkp.nilaiTesKesehatan > l.passingGrade
    ->          THEN 1
    ->          ELSE NULL
    ->          END) as jumlahTesKesehatan,
    -> AVG(ntkp.nilaiTesKesehatan) as ratarata,
    -> MAX(ntkp.nilaiTesKesehatan) as nilaiMaksimum,
    -> MIN(ntkp.nilaiTesKesehatan) as nilaiMinimum
    -> FROM nilaiTesKesehatanPendaftar as ntkp, tesKesehatan as tk, seleksi as s, kelolosan as l;
Query OK, 0 rows affected (0.007 sec)
```

- Query untuk menampilkan informasi

```
SELECT * from dataSeleksiTesKesehatan;
```

```
MariaDB [lpdp2]> SELECT * from dataSeleksiTesKesehatan;
+-------------------+------------+--------------+--------------+
| jumlahTesKesehatan | ratarata   | nilaiMaksimum | nilaiMinimum |
+-------------------+------------+--------------+--------------+
|                 10 | 172.549500 |        383.77 |        16.11 |
+-------------------+------------+--------------+--------------+
1 row in set (0.038 sec)
```

2. Membuat *trigger* untuk mengganti passing grade untuk setiap jenis seleksi menjadi data yang dinamis, yakni rata-rata dari seluruh nilai pada setiap jenis seleksi. Nilai tersebut akan berubah setiap ada perubahan, penambahan, atau pengurangan nilai.

   - Query pembuatan *trigger* untuk mengganti passing grade seleksi wawancara

```
CREATE TRIGGER update_passing_grade_wawancara
AFTER INSERT ON nilaiWawancaraPendaftar
FOR EACH ROW
BEGIN
        DECLARE passingGradeUpdate DECIMAL(5,2);
        SET passingGradeUpdate = (SELECT AVG(nilaiWawancara) FROM
        nilaiWawancaraPendaftar);
        UPDATE Kelolosan
        SET passingGrade = passingGradeUpdate
        WHERE idPass = 1;
END;
```

   - Penjelasan:

Trigger ini dibuat untuk mengganti nilai passing grade pada seleksi wawancara. Trigger dilakukan setelah insert (memasukkan nilai) pada tabel nilaiWawancaraPendaftar. Proses triggernya adalah dengan mendeklarasikan variabel nilai passingGradeUpdate, lalu diset sama dengan rata-rata atribut nilaiWawancara dari tabel nilaiWawancaraPendaftar. Kemudian set nilai passingGrade yang ada pada tabel kelolosan sama dengan variabel passingGradeUpdate yang telah didefinisikan sebelumnya. Nilai passingGrade yang diganti adalah ketika idPassnya = 1 karena ini merupakan idPass untuk seleksi wawancara.

   - Query pembuatan *trigger* untuk mengganti passing grade seleksi psikotest

```
CREATE TRIGGER update_passing_grade_psikotest
AFTER INSERT ON nilaiPsikotestPendaftar
FOR EACH ROW
BEGIN
        DECLARE passingGradeUpdate DECIMAL(5,2);
        SET passingGradeUpdate = (SELECT AVG(nilaiPsikotest) FROM
        nilaiPsikotestPendaftar);
        UPDATE Kelolosan
        SET passingGrade = passingGradeUpdate
        WHERE idPass = 2;
END;
```

- Penjelasan:

Trigger ini dibuat untuk mengganti nilai passing grade pada seleksi psikotest. Trigger dilakukan setelah insert (memasukkan nilai) pada tabel nilaiPsikotestPendaftar. Proses triggernya adalah dengan mendeklarasikan variabel nilai passingGradeUpdate, lalu diset sama dengan rata-rata atribut nilaiPsikotest dari tabel nilaiPsikotestPendaftar. Kemudian set nilai passingGrade yang ada pada tabel kelolosan sama dengan variabel passingGradeUpdate yang telah didefinisikan sebelumnya. Nilai passingGrade yang diganti adalah ketika idPassnya = 2 karena ini merupakan idPass untuk seleksi psikotest.

- Query pembuatan *trigger* untuk mengganti passing grade seleksi tesKesehatan

```
CREATE TRIGGER update_passing_grade_tesKesehatan
AFTER INSERT ON nilaiTesKesehatanPendaftar
FOR EACH ROW
BEGIN
        DECLARE passingGradeUpdate DECIMAL(5,2);
        SET passingGradeUpdate = (SELECT AVG(nilaiTesKesehatan) FROM
        nilaiTesKesehatanPendaftar);
        UPDATE Kelolosan
        SET passingGrade = passingGradeUpdate
        WHERE idPass = 3;
END;
```

- Penjelasan:

Trigger ini dibuat untuk mengganti nilai passing grade pada seleksi tesKesehatan. Trigger dilakukan setelah insert (memasukkan nilai) pada tabel nilaiTesKesehatanPendaftar. Proses triggernya adalah dengan mendeklarasikan variabel nilai passingGradeUpdate, lalu diset sama dengan rata-rata atribut nilaiTesKesehatan dari tabel nilaiTesKesehatanPendaftar. Kemudian set nilai passingGrade yang ada pada tabel kelolosan sama dengan variabel passingGradeUpdate yang telah didefinisikan sebelumnya. Nilai passingGrade yang diganti adalah ketika idPassnya = 3 karena ini merupakan idPass untuk seleksi tesKesehatan.

3. Memastikan setiap pendaftar hanya mengikuti maksimal 3 kali seleksi, dengan tidak ada pendaftar yang mengikuti suatu jenis seleksi lebih dari satu kali dan memastikan juga seleksi hanya bisa dilakukan secara berurutan dimulai dari seleksi wawancara -> seleksi psikotes -> seleksi kesehatan.

- Query pembatasan pendaftar bisa mengikuti tes maksimal 3x

```
ALTER TABLE tesPendaftar
ADD CONSTRAINT cek_maksimal_seleksi CHECK (
        (CASE WHEN idWawancara IS NOT NULL THEN 1 ELSE 0 END +
        CASE WHEN idPsikotest IS NOT NULL THEN 1 ELSE 0 END +
        CASE WHEN idTesKesehatan IS NOT NULL THEN 1 ELSE 0 END) <= 3
);
```

Menambahkan kondisi *constraint* dari tesPendaftar, saat mengikuti seleksi (id tidak NULL, maka menambahkan nilai 1 yang berarti mengikuti 1x tes dan membuat batasan agar nilai kurang dari 3 (hanya bisa mengikuti tes maksimal 3x)

- Query pembatasan pendaftar tidak mengikuti suatu jenis seleksi lebih dari satu kali

```
ALTER TABLE tesPendaftar
ADD CONSTRAINT unique_selections
UNIQUE (idMahasiswa, idWawancara, idTesKesehatan);
```

Menambahkan kondisi *constraint* dari tesPendaftar, saat mengikuti seleksi agar unique saja, tidak boleh ada id yang sama. Untuk psikotest, bisa terdapat kesamaan karena seleksi bisa dilakukan di ruangan yang sama sehingga idPsikotest antar peserta memungkinkan sama sehingga tidak masuk UNIQUE

- Query pembatasan pendaftar bisa mengikuti tes secara berurutan

```
CREATE TABLE tesPendaftar (
   idMahasiswa INT not null PRIMARY KEY,
   idWawancara INT,
   idPsikotest INT,
   idTesKesehatan INT,
   FOREIGN KEY (idMahasiswa) REFERENCES infoPendaftar(idMahasiswa),
   FOREIGN KEY (idPsikotest) REFERENCES nilaiPsikotestPendaftar
(idPsikotest),
   FOREIGN KEY (idTesKesehatan) REFERENCES nilaiTesKesehatanPendaftar
(idTesKesehatan),
   CHECK ((idWawancara IS NOT NULL AND idPsikotest IS NOT NULL) OR
idPsikotest IS NULL),
   CHECK ((idTesKesehatan IS NOT NULL AND idPsikotest IS NOT NULL) OR
idTesKesehatan IS NULL)
);
```

Menambahkan kondisi pengecekan

- Hasil pengecekan

Terjadi perubahan kondisi data dalam database untuk idMahasiswa ke 53 ketika nilai tidak berurutan:

```
736  insert into tesPendaftar (idMahasiswa, idWawancara, idPsikotest, idTesKesehatan) values (52, NULL, NULL, NULL);
737  insert into tesPendaftar (idMahasiswa, idWawancara, idPsikotest, idTesKesehatan) values (53, NULL, NULL, 10);
738  insert into tesPendaftar (idMahasiswa, idWawancara, idPsikotest, idTesKesehatan) values (54, NULL, NULL, NULL);
```

Dalam tabel mariaDB bentuk akhirnya tidak akan dimunculkan:

51	NULL	NULL	NULL
52	NULL	NULL	NULL
54	NULL	NULL	NULL
55	33	62	NULL

BAB III
KESIMPULAN

Kesimpulan yang ada berdasarkan hasil pengerjaan Tugas Besar Basis Data Milestone III ini adalah sebagai berikut:

1. Pemanfaatan normalisasi basis data dapat dilakukan dengan tujuan untuk mempermudah akses dan pengelolaan dari keseluruhan data yang dimiliki. Proses normalisasi data dapat dilakukan dengan menentukan *functional dependencies* dari data relasional. Kemudian menentukan *candidate key* yang dapat terhubung ke semua atribut. Dari bentuk *candidate key* kemudian dapat dicari bentuk normalnya dan dilakukan proses normalisasi hingga data yang ada sudah dalam kondisi BCNF.
2. Basis data yang dimiliki dapat dipergunakan untuk berbagai macam keperluan *business* dan perlu dilakukan penambahan *integrity constraint* untuk membuat data menjadi lebih akurat dan tidak meleset.

REFERENSI

[1] Tim Pengajar IF2140/IF2240 (2020). *Database Design using E-R Model* [Powerpoint Slides].
Available:https://cdn-edunex.itb.ac.id/48686-Database-Parallel-Class/143662-Week-06-SQL/68095-Design-Database-Using-ER-Model/1677186974842_IF2240---SemII_2223---m06-2---Design-using-E-R-Model---part-1.pdf

[2] Tim Pengajar IF2140 Semester I (2020). *Database Design using E-R Model (p.2)* [PowerPoint Slides].
Available:https://cdn-edunex.itb.ac.id/storages/files/1677646554405_IF2240---SemII_2223---m07-1---Design-using-E-R-Model---part-2.pdf

[3] Tim Pengajar IF2140 Semester I (2020). *Database Design using E-R Model (p.3)* [PowerPoint Slides].
Available:https://cdn-edunex.itb.ac.id/storages/files/1678768255399_IF2240---SemII_2223---m09-1---Design-using-E-R-Model---part-3.pdf

[4] Tim Pengajar IF2140/IF2240 (2020). *Relational Database Design* [Powerpoint Slides].
Available:https://cdn-edunex.itb.ac.id/38016-Database-Parallel-Class/71649-Week-10-Reducing-ER-Model-to-Relational-Schema-Relational-Database-Design/39148-Relational-Database-Design-part-1/1648088057587_IF2240---Relational-Database-Design-part-1.pdf

[5] Tim Pengajar IF2140 Semester I (2020). *Relational Database Design (p.2)* [PowerPoint Slides].
Available:https://cdn-edunex.itb.ac.id/38016-Database-Parallel-Class/71650-Week-11-Relational-Database-Design/39385-Relational-Database-Design-part-2/1648462375923_IF2240---Relational-Database-Design-part-2.pdf

[6] Tim Pengajar IF2140 Semester I (2020). *Relational Database Design (p.3)* [PowerPoint Slides].
Available:https://cdn-edunex.itb.ac.id/38016-Database-Parallel-Class/71651-Week-12-Relational-Database-Design-Quiz-2/39835-Relational-Database-Design-part-3/1649387964094_IF2240---Relational-Database-Design-part-3.pdf

[7] Silberschatz, A., Korth, H. F., & Sudarshan, S. (2019). *Database System Concepts", 7th Edition*. McGraw Hill Education.