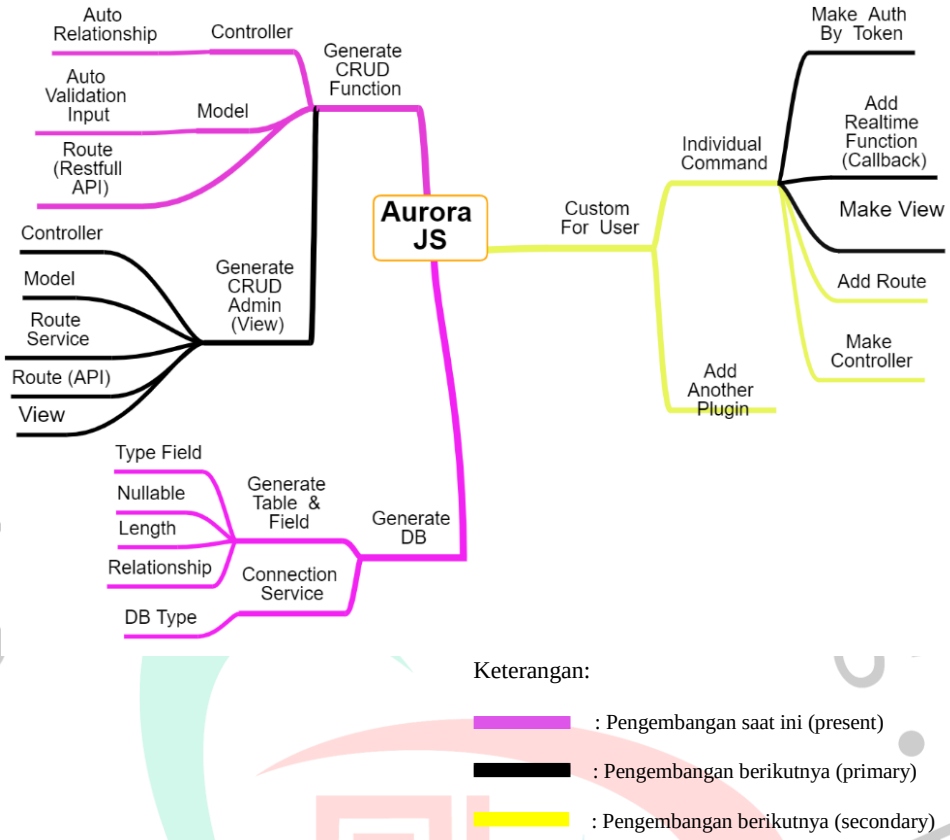
BAB IV
HASIL DAN ANALISIS PENELITIAN

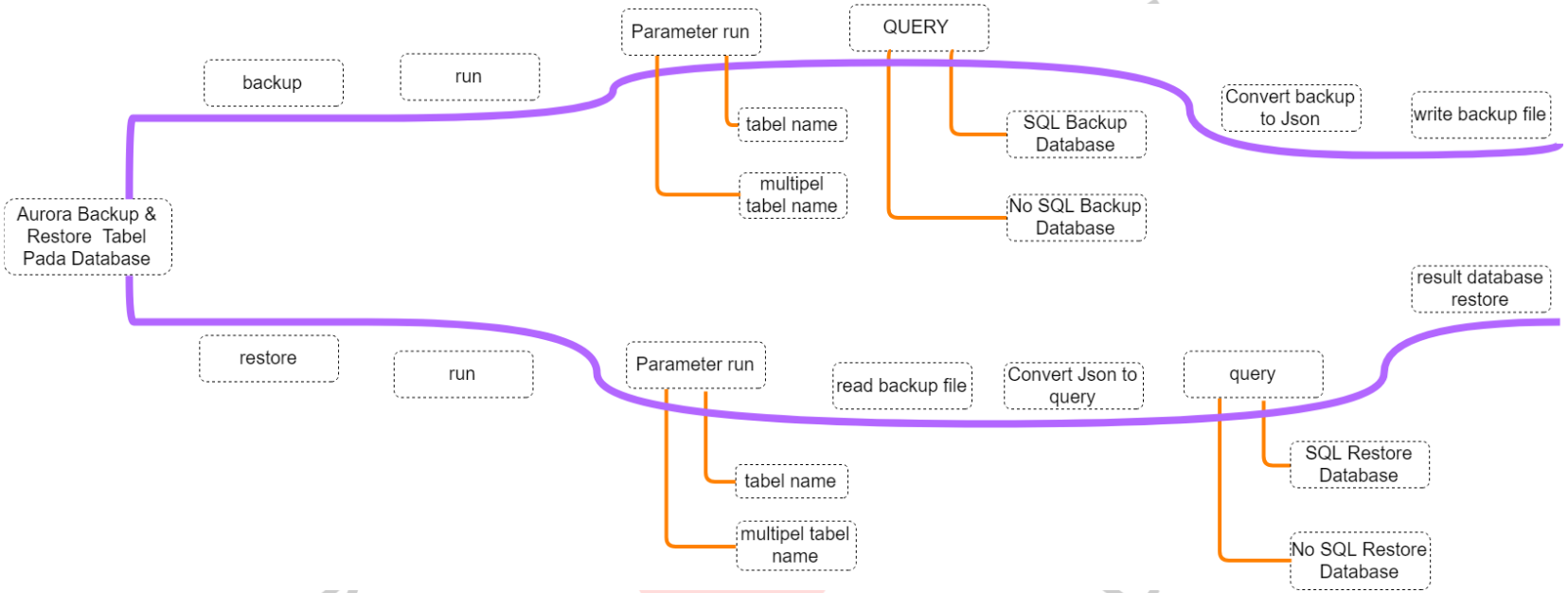
4.1 Analisa Perancangan Sistem

Pengembangan *framework* Aurora.JS berbasis Node.JS yang dikembangkan oleh tim pengembang sekaligus peneliti sebelumnya, dari hal ini didapatkan hasil dari analisis permasalahan, Penulis memberikan solusi dengan tujuan mempermudah pengguna *framework* Aurora.JS dalam mengembangkan aplikasi dengan menambahkan fitur yang dapat mempercepat pekerjaan dalam mengelola data pada *table* database yaitu modul *backup* dan *restore table* pada database tanpa perlu melakukan akses ke database MySQL dan mempermudah pekerjaan tim dalam memanfaatkan data JSON yang sudah di *backup* oleh pengguna dan dibagikan kepada tim melalui github. Perancangan *framework* Aurora.JS yang sudah dirancang oleh peneliti sebelumnya oleh mahasiswa universitas pembangunan jaya prodi sistem informasi yang dikembangkan oleh penulis dengan maksud untuk menyempurnakan hasil produk dengan tujuan menjadikan *framework* Aurora.JS dengan skala besar yang dapat digunakan oleh pengguna maupun pengembang aplikasi untuk mempermudah dan mempercepat waktu pekerjaan. Dalam mengembangkan *framework* Aurora.JS penulis membaca analisa rancangan dari peneliti sebelumnya yaitu *mind mapping* sebagai acuan penulis untuk memahami alur sistem dalam pengembangan Aurora.JS, selanjutnya hasil dari *mind mapping* yang sudah dipahami penulis sebagai dasar acuan penelitian dalam menerapkan fitur yang dikembangkan yaitu fitur *backup* dan *restore table* pada database pada *framework* Aurora.JS berbasis Node.JS, berikut ini *mind mapping* sebelumnya **Gambar 4.1** dan *mind mapping* hasil penelitian yang dirancang dan dibuat pada penulis **Gambar 4.2** :



Gambar 4.1. *Mind Mapping* Aurora.JS

Gambar 4.2. *Mind Mapping* fitur *backup* dan *restore table* pada database Aurora.JS, dari rancangan tersebut merupakan proses yang menjadi dasar dalam melakukan penelitian untuk merancang fitur *backup table* dan *restore table* pada database dengan menggunakan *framework* Aurora.JS, hal ini dilakukan agar dalam melakukan perancangan yang dilakukan penulis menggunakan *mind mapping* untuk mempermudah pengimplementasi sistem yang dikerjakan pada *framework* Aurora.JS. Hal ini sangat membantu penulis untuk menjadi dasar membangun analisis dan sistem yang dibangun, diharapkan dengan adanya fitur ini nantinya dapat mempermudah pemprosesan database yang dilakukan oleh pengguna dan pengembang dengan memanfaatkan *framework* Aurora.JS dalam membangun aplikasi.



Gambar 4.2. *Mind Mapping* Fitur *Backup* Dan *Restore Table* Database Pada Aurora.JS

4.2 Perancangan Diagram Sistem Usulan

Pada tahap ini perancangan yang dilakukan penulis untuk membuat fitur backup table dan restore table pada database dengan framework Aurora.JS menggunakan metode *object-oriented analysis and design*. Pada tahap ini penulis menggunakan tahapan OOAD diagram yaitu : *use case diagram, spesifikasi use case, activity diagram, sequence diagram,* berikut ini dapat dilihat pada penjelasan di bawah ini :

4.2.1 Use Case Diagram

Gambar 4.3 merupakan hasil rancangan *Use Case* diagram yang dilakukan penulis dalam membangun analisis fitur *backup* dan *restore table* pada database, dalam menjalankan proses perintah *backup table* pada database, pengembang dapat melakukan *backup* secara *single table*, maupun *backup* secara *multi table* dengan syarat harus terkoneksi ke database, hal ini dapat dilakukan *backup* secara *custom* koneksi dan tanpa *custom* koneksi. Selanjutnya menjalankan perintah *restore table* pada database, untuk menjalankan *restore table* pada database harus sudah melakukan *backup table* terlebih dahulu pada *framework* Aurora.JS selanjutnya dapat dilakukan *restore table* untuk mengembalikan data pada *table* database yang sudah terkoneksi, untuk melakukan *restore table* dapat dilakukan dengan *restore single table* dan *restore multi table*, dalam melakukan *restore* juga dapat dilakukan *custom* koneksi dan tanpa *custom* koneksi dengan syarat harus terhubung ke database yang sudah di konfigurasi pengembang, dari hal tersebut dihasilkan rancangan pengembangan sistem yang dilakukan penulis dalam merancang fitur *backup table* dan *restore table* pada database dengan menggunakan *framework* Aurora.JS.

Gambar 4.3. *Use Case* Diagram fitur *Backup* Dan *Restore Table* Pada Database Aurora.JS

4.2.2 Spesifikasi Use Case

Berikut ini tahapan pada spesifikasi *use case* yang merupakan hasil penjelasan alur pada kerja atau tahapan dalam melakukan perancangan sistem modul *backup* dan *restore table* database pada *framework* Aurora.JS dapat dilihat pada beberapa tabel bawah ini.

Tabel 4.1. Spesifikasi *Use Case* Membuat Koneksi Database

<table>
<tr><td>Use Case Name</td><td colspan="2">Membuat Koneksi Database</td></tr>
<tr><td>Actors</td><td colspan="2">Pengembang</td></tr>
<tr><td>Trigger</td><td colspan="2">Membuka file config.js</td></tr>
<tr><td>Preconditions</td><td colspan="2">Terdapat file config.js</td></tr>
<tr><td>Postconditions</td><td colspan="2">Framework aurora.js berhasil terhubung dengan Database</td></tr>
<tr><td rowspan="8">Success Scenario</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Melakukan pengisian konfigurasi koneksi ke database</td><td></td></tr>
<tr><td>2. Menyimpan file konfigurasi koneksi ke database</td><td></td></tr>
<tr><td>3. Menjalankan perintah “node app.js” pada CMD/Terminal</td><td></td></tr>
<tr><td></td><td>4. Pengecekan konfigurasi koneksi ke database</td></tr>
<tr><td></td><td>5. Menjalankan koneksi ke database</td></tr>
<tr><td></td><td>6. Memberikan pesan berhasil</td></tr>
<tr><td>7. Menerima pesan berhasil di command-line atau Terminal</td><td></td></tr>
<tr><td rowspan="2">Alternative Scenario</td><td colspan="2">1. Mendeteksi terdapat kesalahan dalam mengisi konfigurasi koneksi ke database</td></tr>
<tr><td colspan="2">1.Mendeteksi terjadi kegagalan dalam melakukan proses koneksi ke database</td></tr>
</table>

Tabel 4.2. Spesifikasi *Use Case* Menjalankan Perintah *Backup*

Use Case Name	Menjalankan Perintah *Backup*	
Actor	Pengembang	
Trigger	Menjalankan perintah *Backup table* di terminal pada *framework* Aurora.JS	
Pre-Condition	Sudah terdapat *table* pada database untuk dilakukan *backup*	
Post-Condition	Bertambah file JSON di folder *backup* pada *framework* Aurora.JS	
Success Scenario	***Actor***	***System***
	1. Pengembang dapat melakukan pemilihan dalam menjalankan perintah *backup single table* atau perintah *backup multi table*	
	2. Menjalankan perintah *backup* database "aurora db:backup –t nama_*table*" pada command line/Terminal. <<include>> menjalankan *backup single table*	
		3. Menjalankan perintah schema *backup single table* pada database
		4. Pengecekan pada *single table* yang akan dilakukan *backup*
		5. Berhasil menemukan *table* untuk di *backup*
		6. *Convert single table* pada database ke *file* JSON di folder *backup*
		7. Memberikan pesan berhasil menjalankan perintah
	8. Menerima pesan di command line/Terminal berhasil menjalankan perintah *backup single table* pada database	

	9. Menjalankan perintah *backup table* pada database "aurora db:*backup* –t nama_*table*, nama_*table*" pada command line/Terminal. <<include>> menjalankan *backup multi table*	
		10. Menjalankan perintah *backup multi table* pada database
		11. Pengecekan pada *multi table* yang akan dilakukan *backup*
		12. Berhasil menemukan *file multi tabel* untuk di *backup*
		13. *Convert multi table* pada database ke *file* JSON di folder *backup*
		14. Memberikan pesan berhasil menjalankan perintah
	15. Menerima pesan di command line/Terminal berhasil menjalankan perintah *backup multi table* pada database	
Alternative Scenario	1&7. Sistem Mendeteksi adanya kesalahan dalam memberikan perintah *backup table* pada command line/Terminal	
	1&7.Gagal menemukan tabel untuk di *backup*.	

Tabel 4.3. Spesifikasi *Use Case* Menjalankan perintah *backup single table*

Use Case Name	Menjalankan perintah *backup single table*
Actor	Pengembang
Trigger	Menjalankan perintah *backup single table* di terminal pada *framework* Aurora.JS
Pre-Condition	Sudah terdapat tabel pada database untuk dilakukan backup

Post-Condition	bertambah file JSON di folder *backup* pada *framework* Aurora.JS	
Success Scenario	***Actor***	***System***
	1. Menjalankan perintah perintah di terminal *backup single table* "aurora db:backup -t nama_*table*" pada CMD/Terminal.	
		2. Menjalankan perintah di terminal *backup single table*
		3. Menjalankan pengecekan *single table* yang ingin dilakukan *backup*
		4. Berhasil menemukan *single table* yang ingin di *backup*
		5. Tabel tersedia berhasil di *convert* menjadi *file* JSON di folder *backup*
		6. Memberikan pesan berhasil menjalankan perintah *backup single table*
	7. Menerima pesan di command line/Terminal berhasil menjalankan perintah *backup single tabel* pada database	
	8. *If* Menjalankan " aurora db:backup " atau "aurora db:backup -schema nama *file* schema" <<extend membuat backup single table pada database tanpa *custom* koneksi>> selection menjalankan "aurora db:backup –s" nama *file custom* koneksi" atau " aurora db:backup -schema / -s "nama *file* schema" nama *file custom* koneksi <<extend membuat backup single table pada database dengan *custom* koneksi>>	

Alternative Scenario	1. Sistem mendeteksi adanya kesalahan dalam memberikan perintah koneksi schema *backup single table* pada command line/Terminal.
	7. Sistem mendeteksi adanya kesalahan dalam memberikan perintah nama *single table* yang ingin di *backup*

Tabel 4.4. Spesifikasi *Use Case* Membuat *Backup Single Table* tanpa *Custom* Koneksi

Use Case Name	Membuat *Backup Single Table* tanpa *Custom* Koneksi	
Actor	Pengembang	
Trigger	Menjalankan perintah *backup single table* tanpa *custom* koneksi pada database di *command line*/Terminal	
Pre-Condition	Menjalankan perintah pembuatan *backup single table* tanpa *custom* koneksi pada *command line*/Terminal	
Post-Condition	*File backup* berhasil dibuat pada database sesuai dengan konfigurasi koneksi pada *file default* koneksi	
Success Scenario	***Actor***	***System***
	1. <<include membuat koneksi database>>	
	2. Menjalankan perintah “aurora db:backup -t nama_tabel, nama_tabel –s” “nama_file_schema” nama file koneksi pada command line/Terminal	
		3. Pengecekan file schema
		4. Memeriksa konfigurasi pada variabel module.exsport.backup
		5. Pembuatan syntax SQL tanpa *custom* koneksi
		6. Menjalankan konfigurasi koneksi pada database dari *file default* koneksi

		7. Menjalankan syntax SQL tanpa *custom* koneksi
		8. Memberi pesan tabel berhasil di *backup*
	9. Menerima pesan berhasil di command line/Terminal	
Alternative Scenario	2.File schema tidak ditemukan	
	2. sistem mendeteksi adanya kesalahan dalam konfigurasi pada variabel module.exports.backup	

Tabel 4.5. Spesifikasi *Use Case* Membuat *Backup Single Table* dengan *Custom* Koneksi

Use Case Name	Membuat *Backup Single Table* dengan *Custom* Koneksi	
Actor	Pengembang	
Trigger	Menjalankan perintah *backup single table* dengan *custom* koneksi pada database di *command line* atau Terminal	
Pre-Condition	Menjalankan perintah perubahan *backup single table* dengan *custom* koneksi pada *command line* atau terminal	
Post-Condition	File *backup* berhasil dibuat pada database sesuai dengan konfigurasi *custom* koneksi	
Success Scenario	***Actor***	***System***
	1. <<include membuat koneksi database>>	
	2. Menjalankan perintah costum koneksi “aurora db:backup –t nama_tabel –s” nama_file_schema” nama file koneksi” pada CMD/Terminal	
		3. pengecekan file schema
		4. memeriksa konfigurasi pada variabel module.exports.backup

		5. pembuatan syntax SQL tanpa custom koneksi
		6. Menjalankan konfigurasi koneksi pada database dari *file custom* koneksi
		7. Menjalankan syntax sql dengan costum koneksi
		8. memberi pesan tabel berhasil di *backup*
	9. menerima pesan backup tabel custom koneksi di command line/Terminal berhasil	
	1. *file custom* koneksi atau schema tidak tersedia	
	1. sistem mendeteksi adanya kesalahan pada variabel module.exports.backup	

Tabel 4.6. Spesifikasi *Use Case* Menjalankan Perintah *Backup* multi *table*

Use Case Name	Menjalankan Perintah *backup multi table*	
Actor	Pengembang	
Trigger	Menjalankan perintah *backup multi table* pada *framework* Aurora.JS	
Pre-Condition	Sudah terdapat tabel pada database untuk dilakukan *backup*	
Post-Condition	Penambahan file JSON di folder backup pada *framework* Aurora.JS	
Success Scenario	***Actor***	***System***
	1. Menjalankan perintah perintah *backup multi table* "aurora db:*backup* -t nama_*table*, nama_*table*" pada command line/Terminal	
		2. Menjalankan perintah schema *backup multi table* pada database
		3. Menjalankan pengecekan *multi table*

		yang ingin dilakukan backup
		4. Berhasil menemukan *multi table* yang ingin di backup
		5. Tabel tersedia berhasil di convert menjadi file JSON di folder *backup*
		6. Memberikan pesan berhasil menjalankan perintah *backup multi table*
	7. Menerima pesan di command line/Terminal berhasil menjalankan perintah *backup multi table* pada database	
	8. If menjalankan " aurora db:backup" atau "aurora db:backup nama_table, nama_table nama_*file*_schema/nama file koneksi" <<extend membuat tabel pada database tanpa *custom* koneksi>> selection menjalankan "aurora db:backup nama_table,nama_table -s" atau " aurora db:backup -s nama *file* schema/nama *file custom* koneksi" <<extend membuat tabel pada database dengan *custom* koneksi>>	
Alternative Scenario	1.Sistem Mendeteksi adanya kesalahan dalam memberikan perintah koneksi schema *backup multi table* pada command line/Terminal.	
	7.Sistem mendeteksi adanya kesalahan dalam memberikan perintah nama *multi table* yang ingin di *backup*	

Tabel 4.7. Spesifikasi *Use Case* Membuat *Backup Multi Table* tanpa *Custom* Koneksi

Use Case Name	Membuat Backup *Multi Table* tanpa *Custom* Koneksi
Actor	Pengembang
Trigger	Menjalankan perintah *backup multi table* tanpa *custom* koneksi pada database di *command line*/Terminal

Pre-Condition	Menjalankan perintah pembuatan *backup multi table* tanpa *custom* koneksi pada *command line*/Terminal	
Post-Condition	*File backup* berhasil dibuat pada database sesuai dengan konfigurasi koneksi pada file default koneksi	
Success Scenario	***Actor***	***System***
	1. <<include membuat koneksi database>>	
	2. menjalankan perintah "aurora db:backup -t nama_tabel –s" nama_file_schema" nama_file_koneksi" pada command line/Terminal	
		3. pengecekan file schema
		4. memeriksa konfigurasi pada variabel module.exports.backup
		5. pembuatan syntax SQL tanpa custom koneksi
		6. menjalankan konfigurasi koneksi pada database dari file default koneksi
		7. menjalankan syntax sql tanpa custom koneksi
		8. memberi pesan tabel berhasil di backup
	9. menerima pesan berhasil di command line/Terminal.	
Alternative Scenario	1. sistem mendeteksi adanya kesalahan dalam konfigurasi pada variabel module.exports.backup	
	2. *file custom* koneksi atau schema tidak tersedia	
	2.sistem mendeteksi adanya kesalahan dalam konfigurasi pada variabel module.exports.backup	

Tabel 4.8. Spesifikasi *Use Case* Membuat *Backup Multi Table* dengan *Custom* Koneksi

Use Case Name	Membuat *Backup Multi Table* dengan *Custom* Koneksi	
Actor	Pengembang	
Trigger	Menjalankan perintah *backup multi table* dengan *custom* koneksi pada database di *command line* atau Terminal	
Pre-Condition	Menjalankan perintah perubahan *backup multi table* dengan *custom* koneksi pada *command line* atau terminal	
Post-Condition	*File backup* berhasil dibuat pada database sesuai dengan konfigurasi *custom* koneksi	
Success Scenario	***Actor***	***System***
	1. <<include membuat koneksi database>>	
	2. menjalankan perintah *costum* koneksi "aurora db:backup –t nama_tabel, nama_tabel –s" nama_file_schema" nama_file_koneksi" pada command line/Terminal	
		3. pengecekan file schema
		4. memeriksa konfigurasi pada variabel module.exports.backup
		5. pembuatan syntax SQL tanpa *custom* koneksi
		6. Menjalankan konfigurasi koneksi pada database dari *file custom* koneksi
		7. Menjalankan syntax sql dengan costum koneksi
		8. memberi pesan tabel berhasil di *backup*
	9. menerima pesan *backup* tabel *custom* koneksi di *command line*/Terminal berhasil	

Alternative Scenario	1. sistem mendeteksi kesalahan dalam proses konfigurasi koneksi ke database
	2.sistem mendeteksi adanya kesalahan dalam konfigurasi pada variabel module.exports.backup
	2. *file custom* koneksi atau schema tidak tersedia

Tabel 4.9. Spesifikasi *Use Case* Menjalankan Perintah *restore*

Use Case Name	Menjalankan Perintah *restore*	
Actor	Pengembang	
Trigger	Menjalankan perintah *restore table* di terminal pada *framework* Aurora.JS	
Pre-Condition	sudah terdapat *file* JSON yang sudah di *backup* pada folder *backup*	
Post-Condition	*File* JSON yang terdapat di folder *backup* sudah di *restore* di database	
Success Scenario	***Actor***	***System***
	1. Menjalankan perintah schema *restore* database “aurora db:*restore* –t nama_*table*,nama_*table*” pada command line/Terminal. <<include>> menjalankan schema *restore single table*	
		2. Menjalankan perintah schema *restore single table* pada database
		3. Pengecekan pada folder *backup* yang akan dilakukan *single restore*
		4. Berhasil menemukan *single table* untuk di *restore*
		5. *Read file* JSON di folder *backup* untuk di *restore table* pada database

		6. Memberikan pesan berhasil menjalankan perintah
	7. Menerima pesan di command line/Terminal berhasil menjalankan perintah *restore single table* pada database	
	8. Menjalankan perintah schema *restore* database "aurora db:restore –t nama_*table*, nama_*table*" pada command line/Terminal. <<include>> menjalankan schema *restore multi table*	
		9. Menjalankan perintah schema *restore multi table* pada database
		10. Pengecekan pada *multi table* yang terdapat di folder *backup* yang akan dilakukan *multi restore*
		11. Berhasil menemukan *multi table* untuk di *restore*
		12. *Read file* JSON di folder *backup* untuk di *restore table* pada database
		13. Memberikan pesan berhasil menjalankan perintah
	14. Menerima pesan di command line/Terminal berhasil menjalankan perintah *restore multi table* pada database	
Alternative Scenario	1.sistem mendeteksi kesalahan dalam proses konfigurasi koneksi ke database	
	1 &8. Sistem mendeteksi adanya kesalahan dalam memberikan perintah *restore* tabel pada *command line*/Terminal	
	1&8. Gagal menemukan tabel untuk di restore	

Tabel 4.10. Spesifikasi *Use Case* Menjalankan *restore single table*

Use Case Name	Menjalankan *restore single table*	
Actor	Pengembang	
Trigger	Menjalankan perintah *restore single table* di terminal pada *framework* Aurora.JS	
Pre-Condition	sudah terdapat *file* JSON yang sudah di *backup* pada *framework* Aurora.JS	
Post-Condition	*Restore* data *table* ke dalam database	
Success Scenario	***Actor***	***System***
	1. Menjalankan perintah perintah restore single table "aurora db:restore -t nama_*table*" pada command line/Terminal	
		2. Menjalankan perintah schema *restore single table* pada database
		3. Menjalankan pengecekan pada *table backup* yang terdapat di folder *backup* dan akan dilakukan *restore*
		4. Berhasil menemukan *single tabel* yang ingin di *restore*
		5. *Read file* JSON di folder *backup* untuk di *restore table* pada database
		6. Memberikan pesan berhasil menjalankan perintah restore single table
	7. Menerima pesan di command line/Terminal berhasil menjalankan perintah *restore single table* pada database	

	8. If menjalankan " aurora db:restore" atau "aurora db:restore -s nama_*file*_schema" <<extend restore single table pada database tanpa *custom* koneksi>> selection menjalankan "aurora db:restore nama_*file_custom*_koneksi" atau " aurora db:restore -s nama_*file*_schema" nama_*file_custom*_koneksi" <<extend restore single table pada database dengan *custom* koneksi>>	
Alternative Scenario	1.Sistem mendeteksi adanya kesalahan dalam memberikan perintah schema atau koneksi *restore* tabel pada *command line*/Terminal.	
	8.Sistem mendeteksi adanya kesalahan dalam memberikan perintah koneksi *restore file* schema	

Tabel 4.11. Deskripsi *Use Case Read Restore Single Table* tanpa *Custom* Koneksi

Use Case Name	*Read Restore Single Table* tanpa *Custom* Koneksi	
Actor	Pengembang	
Trigger	Menjalankan perintah *restore single table* tanpa *custom* koneksi pada *framework* Aurora.JS di *command line*/Terminal.	
Pre-Condition	sudah terdapat *file* JSON yang sudah di *backup* pada *framework* Aurora.JS	
Post-Condition	*Restore table* berhasil di input ke database sesuai dengan konfigurasi pada *file default* koneksi	
Success Scenario	***Actor***	***System***
	1. <<include membuat koneksi database>>	
	2. menjalankan perintah "aurora db:restore –t nama_tabel –s" nama_file_schema" nama file koneksi pada command line/terminal	

		3. pengecekan file schema
		4. memeriksa konfigurasi pada variabel module.exports.restore
		5. pembuatan syntax SQL tanpa custom koneksi
		6. menjalankan konfigurasi koneksi pada database dari file default koneksi
		7. menjalankan syntax SQL tanpa custom koneksi
		8. memberi pesan tabel berhasil di restore
	9. menerima pesan berhasil di command line/terminal	
Alternative Scenario	1.sistem mendeteksi kesalahan dalam proses konfigurasi koneksi ke database	
	2.tidak ditemukan file schema atau koneksi	
	2. sistem mendeteksi adanya kesalahan dalam konfigurasi pada variabel module.exports.restore	

Tabel 4.12. Deskripsi *Use Case Read Restore Single Table* dengan *Custom* Koneksi

Use Case Name	*Read Restore Single Table* dengan *Custom* Koneksi	
Actor	Pengembang	
Trigger	Menjalankan perintah *restore single table* dengan *custom* koneksi pada *framework* Aurora.JS di *command line*/Terminal.	
Pre-Condition	sudah terdapat *file* JSON yang sudah di *backup* pada *framework* Aurora.JS	
Post-Condition	*Restore table* berhasil di input ke database sesuai dengan konfigurasi pada *file custom* koneksi	
	Actor	***System***

Success Scenario	1. <<include membuat koneksi database>>	
	2. Menjalankan perintah "aurora db:restore –t nama_tabel –s" nama_file_schema" nama file koneksi pada command line/terminal	
		3. pengecekan file schema
		4. memeriksa konfigurasi pada variabel module.exports.restore
		5. pembuatan syntax SQL
		6. Menjalankan konfigurasi koneksi ke database dari *file custom* koneksi
		7. menjalankan syntax SQL dengan custom koneksi
		8. memberi pesan tabel berhasil di *restore*
	9. menerima pesan *restore custom* koneksi di *command line*/Terminal berhasil	
Alternative Scenario	1. sistem mendeteksi kesalahan dalam konfigurasi *custom* koneksi	
	2.sistem tidak menemukan file schema atau koneksi	
	2.sistem mendeteksi kesalahan dalam konfigurasi pada variabel module.exsport.restore	

Tabel 4.13. Spesifikasi *Use Case* Menjalankan Perintah *Restore multi table*

Use Case Name	Menjalankan perintah *restore multi table*
Actor	Pengembang
Trigger	Menjalankan perintah *restore single table* di terminal pada *framework* Aurora.JS

Pre-Condition	sudah terdapat *file* JSON yang sudah di *backup* pada *framework* Aurora.JS	
Post-Condition	*Restore* data *multi table* ke dalam database	
Success Scenario	***Actor***	***System***
	1. Menjalankan perintah perintah restore multi table “aurora db:restore -t nama_*table*, nama_*table*” pada command line/Terminal.	
		2. Menjalankan perintah schema *restore multi table* pada database
		3. Menjalankan pengecekan pada *table* yang terdapat di folder *backup* dan akan dilakukan *restore*
		4. Berhasil menemukan *multi table* yang ingin di *restore*
		5. *Read file* JSON di folder *backup* untuk di *restore table* pada database
		6. Memberikan pesan berhasil menjalankan perintah *restore multi table*
	7. Menerima pesan di command line/Terminal berhasil menjalankan perintah restore multi tabel pada database	
	8. If menjalankan “ aurora db:restore” atau “aurora db:restore -s nama_*file*_schema” <<extend *restore table* pada database tanpa *custom* koneksi>> Selection menjalankan “aurora db:restore nama_*file_custom*_koneksi” atau “ aurora db:*restore* -s nama_*file*_schema” nama_*file_custom*_koneksi”	

	<<extend *restore table* pada database dengan *custom* koneksi>>	
Alternative Scenario	1. Sistem mendeteksi adanya kesalahan dalam memberikan perintah schema *restore multi table* pada command line/Terminal	
	7. Sistem mendeteksi adanya kesalahan dalam memberikan perintah nama *multi table* yang ingin di *restore*	
	7.Memberikan Pesan gagal koneksi restore *table* pada database	

Tabel 4.14. Deskripsi *Use Case read restore Multi Table* Tanpa *Custom* Koneksi

Use Case Name	*read restore Multi Table* Tanpa *Custom* Koneksi	
Actor	Pengembang	
Trigger	Menjalankan perintah *restore multi table* tanpa *custom* koneksi pada *framework* Aurora.JS di *command line*/Terminal	
Pre-Condition	Terdapat perintah pengembalian data *restore multi table* tanpa *custom* koneksi pada *command line*/Terminal	
Post-Condition	R*estore table* berhasil dii nput ke database sesuai dengan konfigurasi pada *file default* koneksi	
	Actor	***System***
	1. <<include membuat koneksi database>>	
	2. menjalankan perintah “aurora db:restore –t nama_tabel, nama_tabel –s” nama_file_schema” nama file koneksi pada command line/terminal	
		3. pengecekan file schema
		4. memeriksa konfigurasi pada variabel module.exports.restore

Success Scenario		5. pembuatan syntax sql tanpa custom koneksi
		6. menjalankan konfigurasi koneksi pada database dari file default koneksi
		7. menjalankan syntax sql tanpa custom koneksi
		8. memberi pesan tabel berhasil di restore
	9. menerima pesan berhasil di command line/terminal	
Alternative Scenario	1.sistem mendeteksi kesalahan dalam proses konfigurasi koneksi ke database	
	2.tidak ditemukan *file* schema atau koneksi	
	2 sistem mendeteksi adanya kesalahan dalam konfigurasi pada variabel module.exports.restore	

Tabel 4.15. Spesifikasi *Use Case Read Restore Multi Table* dengan *Custom* Koneksi

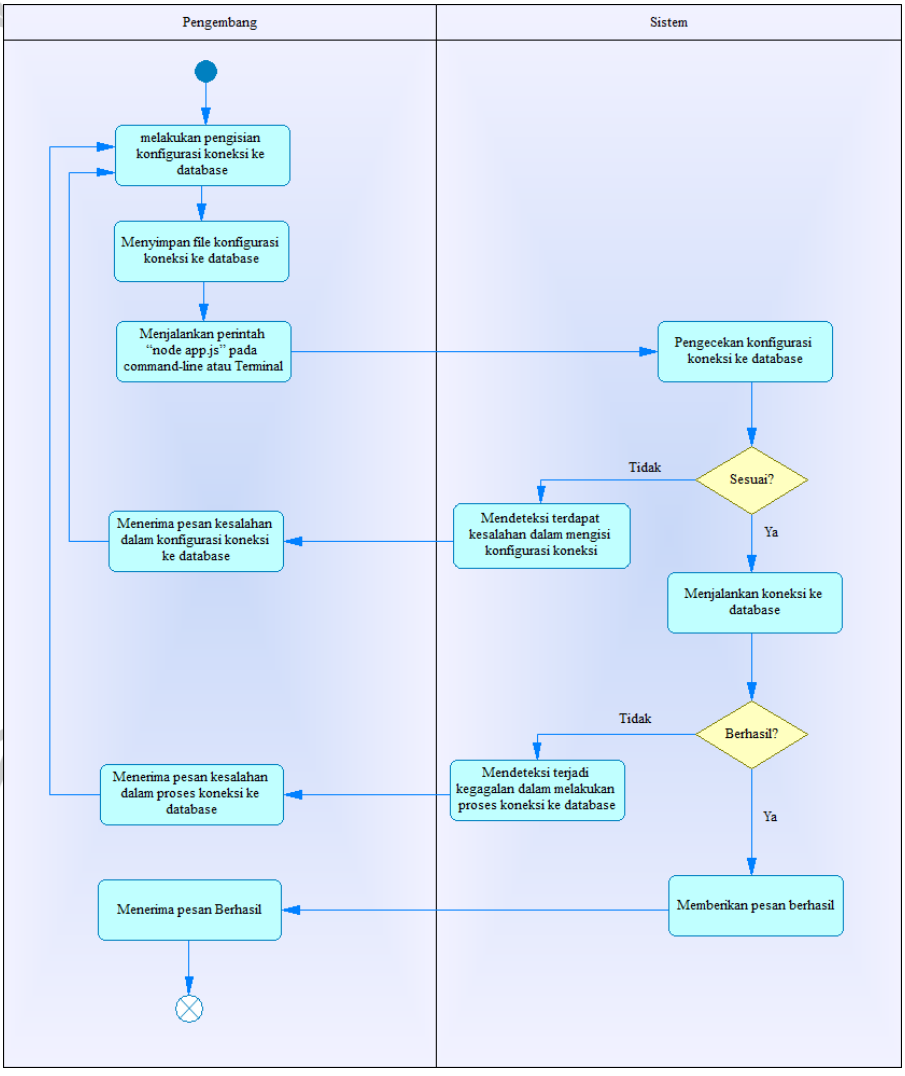
Use Case Name	*Read Restore Multi Table* Dengan *Custom* Koneksi	
Actor	Pengembang	
Trigger	Menjalankan perintah *restore multi table* dengan *custom* koneksi pada *framework* Aurora.JS di *command line*/Terminal.	
Pre-Condition	Terdapat perintah perubahan *restore multi table* tanpa *custom* koneksi pada *framework* Aurora.JS di CMD/Terminal	
Post-Condition	*Restore table* berhasil di input ke database sesuai dengan konfigurasi pada *file custom* koneksi	
Success Scenario	***Actor***	***System***
	1. <<include membuat koneksi database>>	
	2. menjalankan perintah “aurora db:restore –t nama_tabel, nama_tabel –s” “nama_file_schema”	

	nama file koneksi pada command line/terminal	
		3. pengecekan file schema
		4. memeriksa konfigurasi pada variabel module.exports.restore
		5. pembuatan syntax SQL dengan custom koneksi
		6. Menjalankan konfigurasi koneksi ke database dari *file custom* koneksi
		7. menjalankan syntax SQL dengan custom koneksi
		8. memberi pesan tabel berhasil di restore
	9. menerima pesan custom koneksi di command line/Terminal berhasil	
Alternative Scenario	1.sistem mendeteksi kesalahan dalam proses konfigurasi koneksi ke database	
	2.sistem tidak menemukan file schema atau koneksi	
	2.sistem mendeteksi kesalahan dalam konfigurasi pada variabel module.exsport.restore	

4.2.3 Activity Diagram

Activity Diagram yang dirancang penulis menjelaskan alur kerja sistem pada suatu proses atau aktivitas yang dijalankan aktor, dengan adanya *activity diagram* membantu proses perancangan sistem agar lebih mudah dipahami. Dalam hal ini penulis merancang *activity diagram* berdasarkan pada spesifikasi *use case* yang dikerjakan pada penulis.

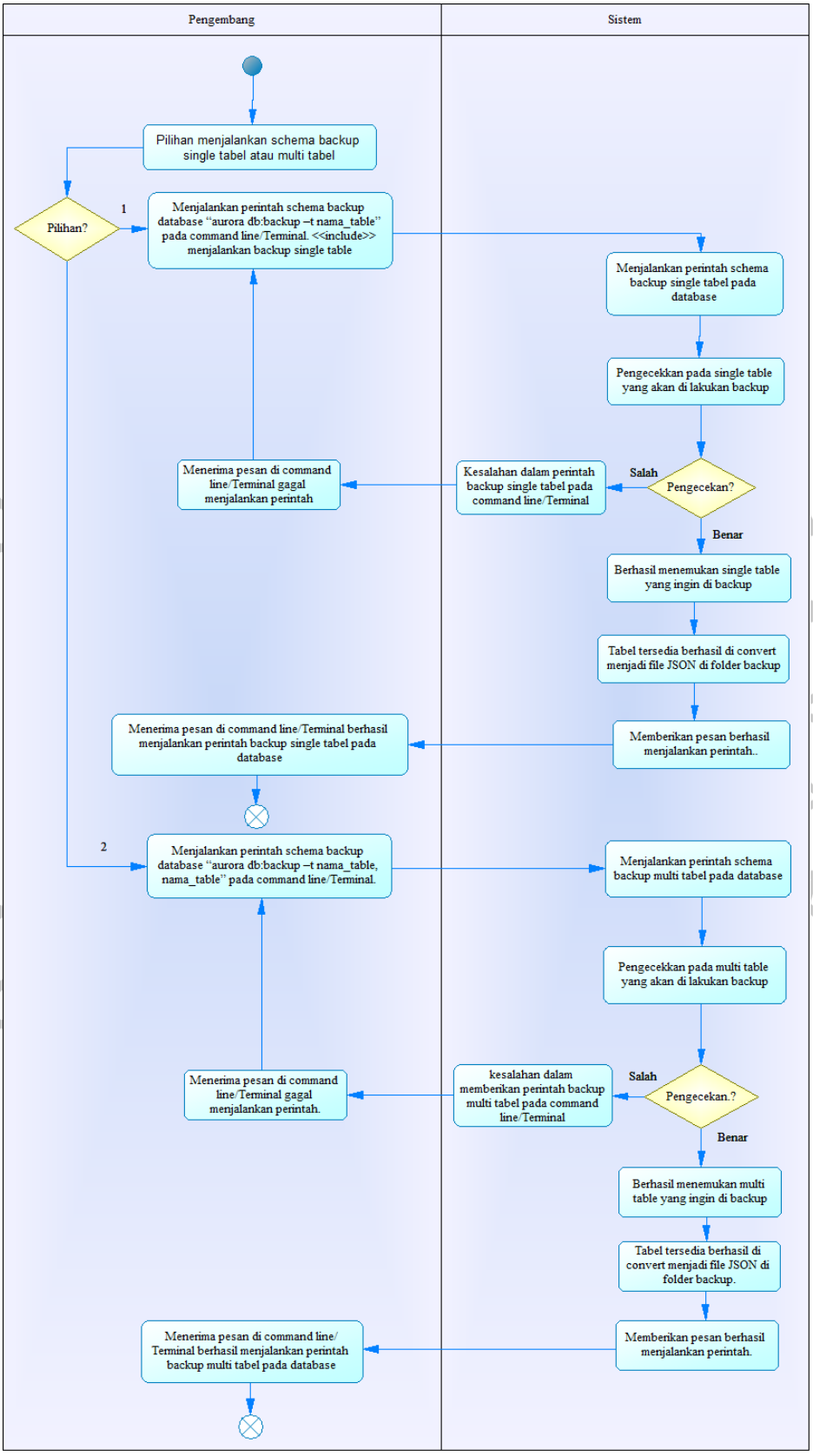
Pada Gambar 4.4. Menjelaskan proses *activity diagram* cara untuk menjalankan suatu proses dalam menghubungkan koneksi ke database melalui *framework* Aurora.JS, hal yang harus dilakukan dengan mengisi konfigurasi koneksi dan menjalankan perintah di *command- line* atau terminal, jika konfigurasi terkoneksi sistem akan menghubungkan koneksi ke database, jika salah terdapat informasi kesalahan pada *command line* konfigurasi terdapat kesalahan. Pada proses ini harus dilakukan koneksi terlebih dahulu dalam menggunakan *fitur backup* dan *restore table*.



Gambar 4.4. *Activity Diagram* Membuat Koneksi Database

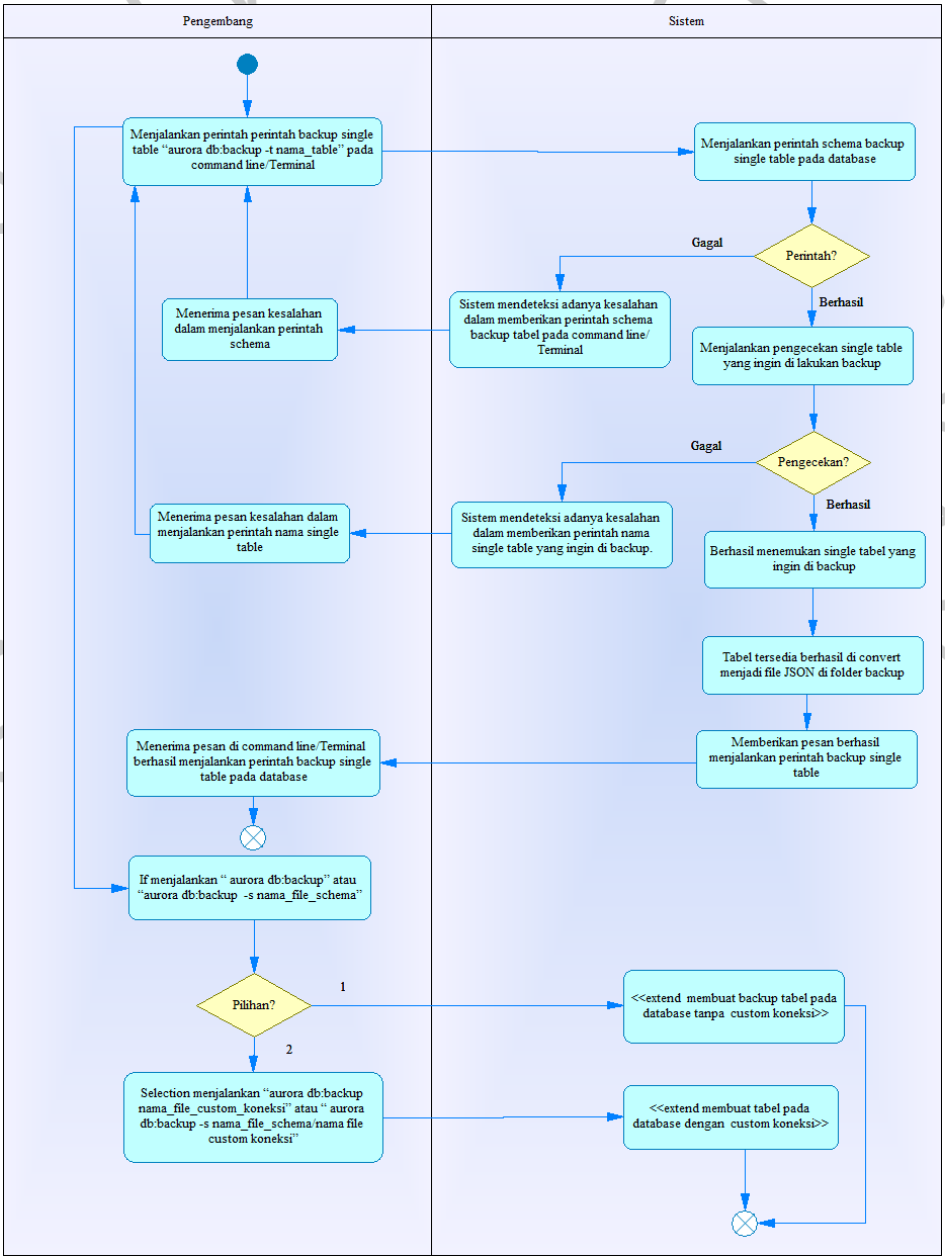
Dapat Dilihat Pada **Gambar 4.5.** Menjelaskan tentang cara menjalankan perintah *backup table* pada database pada *framework* Aurora.JS. dalam menjalankan proses *backup table* pada database terdapat pilihan untuk melakukan *backup single table* atau *backup multi table*, jika proses telah berhasil dijalankan sistem akan melakukan *backup table* pada database yang dirancang dan memberikan informasi di *command line* (CMD), jika perintah *backup* terjadi kesalahan akan terdapat pesan di *command line* (CMD).

Dalam menjalankan perintah *backup single table* dan *backup multi table* dapat ditentukan dengan menambahkan nama *table* pada database di *command line framework* Aurora.JS, jika pengembang ingin melakukan penambahan nama perintah sesuai dengan apa yang diinginkan pengembang atau melakukan penamaan tanpa *custom* koneksi maupun dengan *custom* koneksi dapat dilakukan dengan menambah kalimat di akhir CMD, setelah menjalankan perintah di CMD akan ada penambahan data yang di *backup* di folder *backup* pada *framework* Aurora.JS format akan tersimpan dalam bentuk *file* JSON, jika terjadi kesalahan pengembang harus melakukan perbaikan dalam menjalankan perintah *backup* agar sistem dapat membaca perintah dengan sesuai pada *framework* Aurora.JS. Berikut ini penjelasan *activity diagram* yang dirancang penulis.



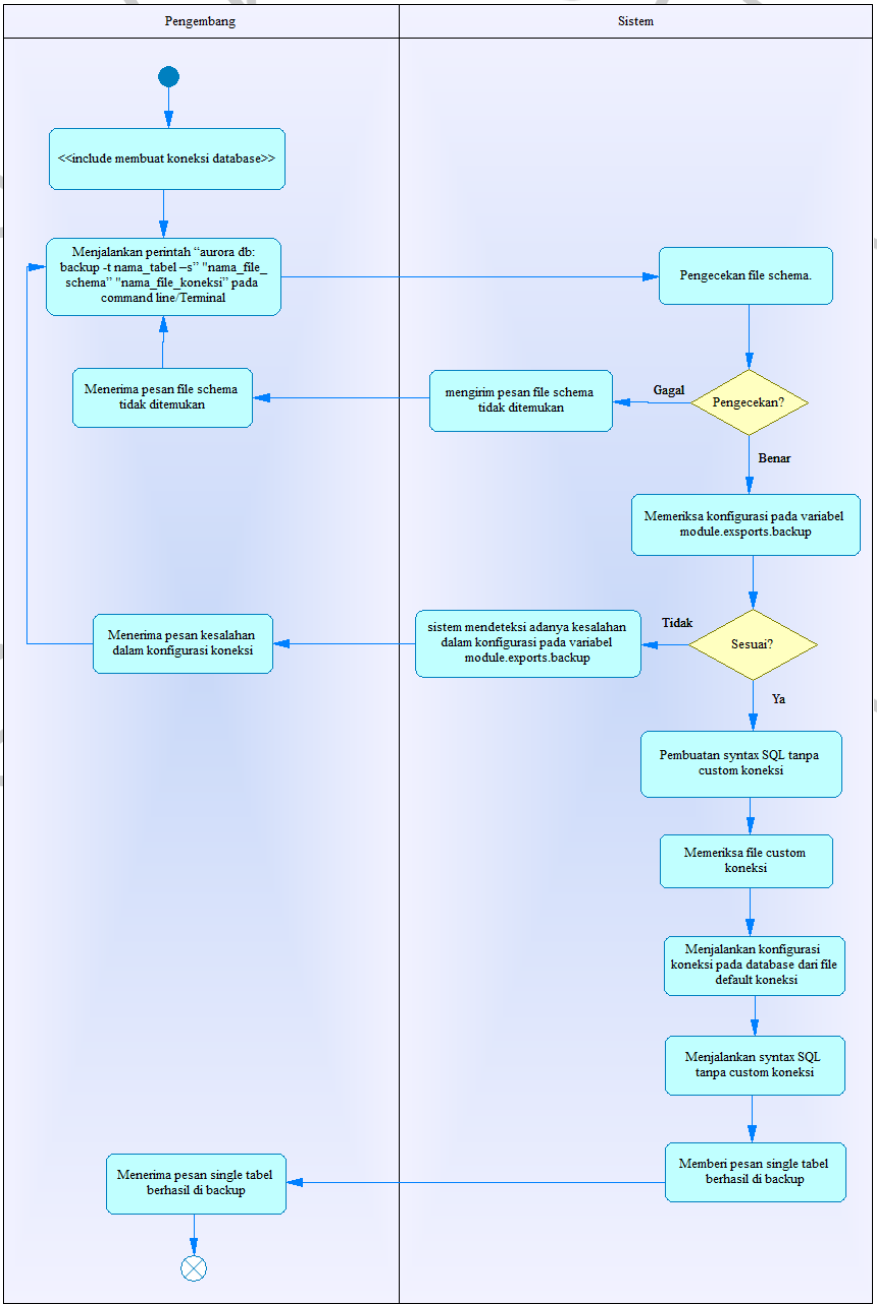
Gambar 4.5. *Activity Diagram* Menjalankan Perintah *Backup*

Pada **Gambar 4.6** merupakan penjelasan mengenai proses yang dijalankan pengembang dan *feedback* yang dilakukan sistem untuk memproses perintah yang dilakukan pengembang, pada tahap ini untuk menjalankan backup single table yang dilakukan pengembang harus sudah melakukan koneksi ke database setelah selesai menjalankan perintah terdapat pesan di *command line* berhasil melakukan *backup single table* yang sudah ditentukan pengembang pada *framework* Aurora.JS.



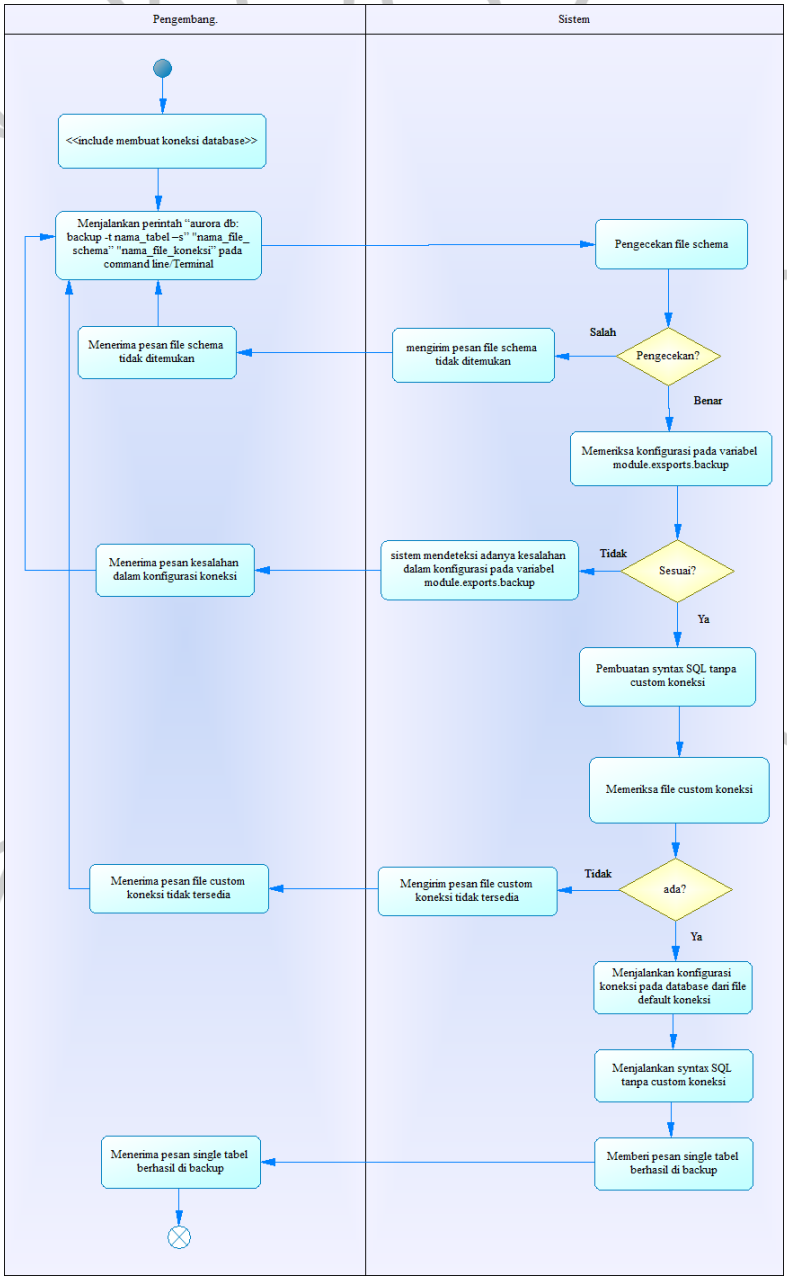
Gambar 4.6. *Activity Diagram* menjalankan Perintah *Backup Single Table*

Pada **Gambar 4.7** merupakan penjelasan activity diagram dalam menjalankan perintah single table tanpa custom koneksi, pada tahap ini jika ingin melakukan default koneksi pengembang sudah melakukan konfigurasi koneksi tanpa adanya penambahan penamaan atau sudah bawaan *file* koneksi dalam melakukan koneksi ke database, hal ini agar *framework* Aurora.JS dapat membaca bahwa *file* koneksi tersebut merupakan *default* yang sudah ditentukan pada *framework* Aurora.JS.



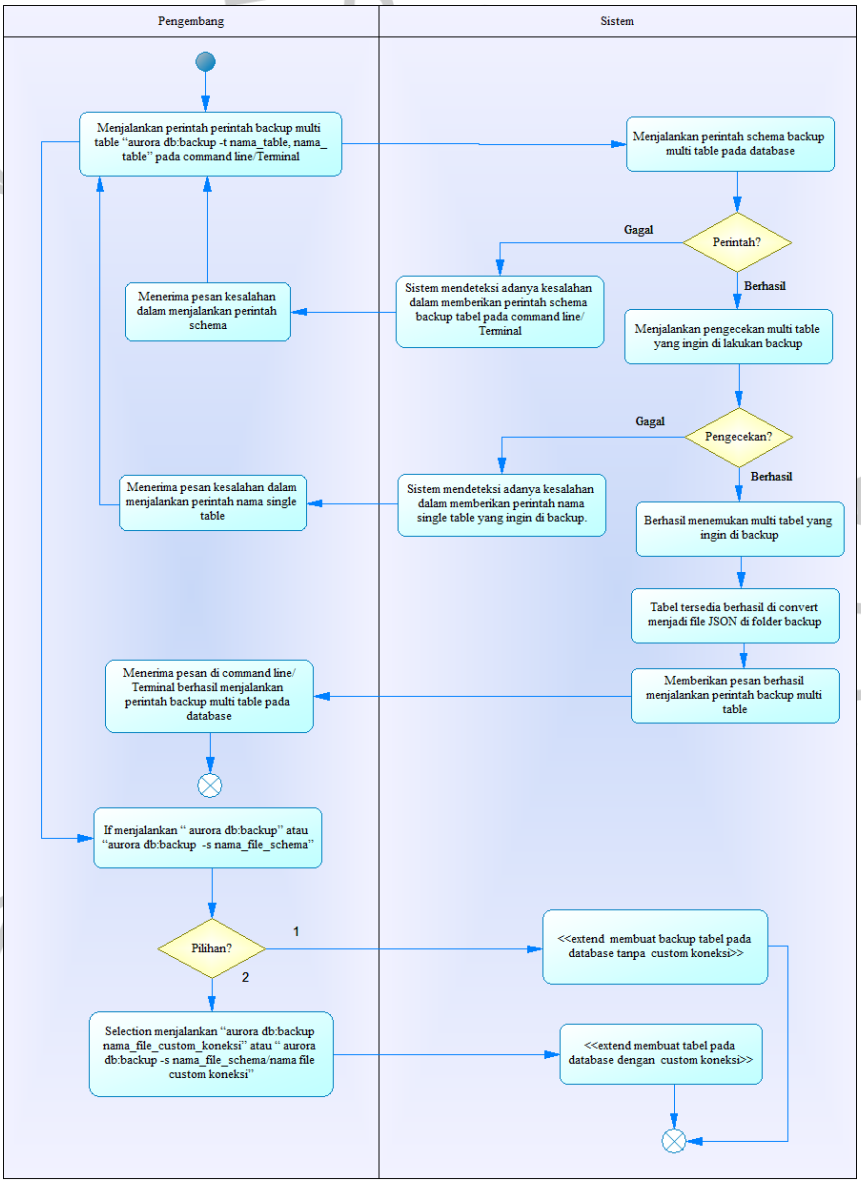
Gambar 4.7. *Activity Diagram* Membuat *backup single table* Tanpa *Custom* Koneksi

Pada **Gambar 4.8** menjelaskan proses menjalankan *backup single* dengan *custom* koneksi, hal ini agar mempermudah pengembang dan pengguna dalam menjalankan koneksi yang ditentukan, dengan adanya *custom* koneksi akan lebih mempermudah penamaan yang diinginkan, dengan menjalankan *custom* koneksi pengembang dapat menambahkan nama koneksi yang diinginkan dengan menambahkan perintah penamaan di belakang pada *command line*.



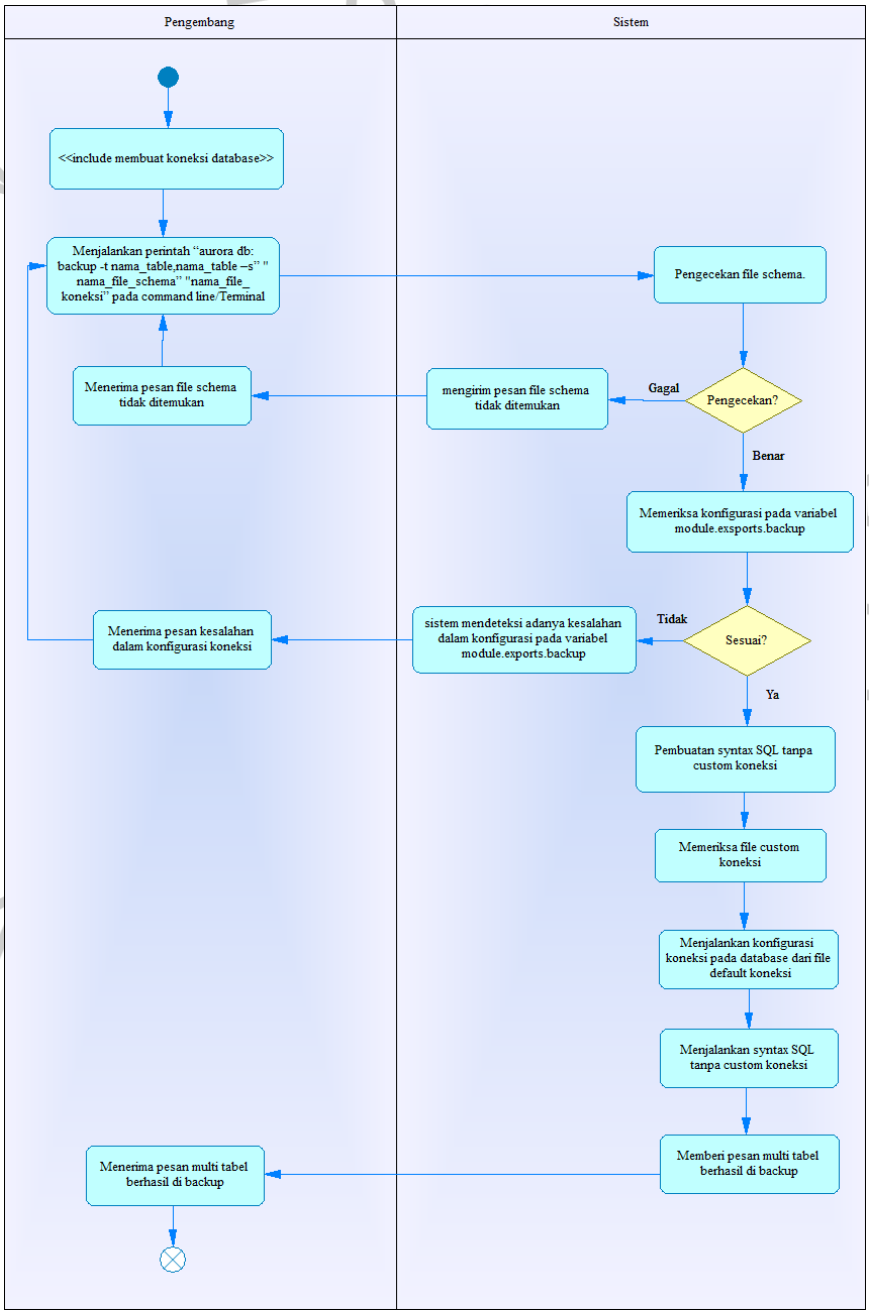
Gambar 4.8. *Activity Diagram* Membuat *backup single table* Dengan *Custom* Koneksi

Pada Gambar 4.9 merupakan proses perintah menjalankan *backup multi table*, untuk menjalankan fitur *backup multi table* pengembang dapat menambahkan nama *table* pada *command line framework* Aurora.JS, untuk mengetahui proses menjalankan perintah dapat dilihat pada gambar tersebut untuk menjalankan perintah *backup multi table.*



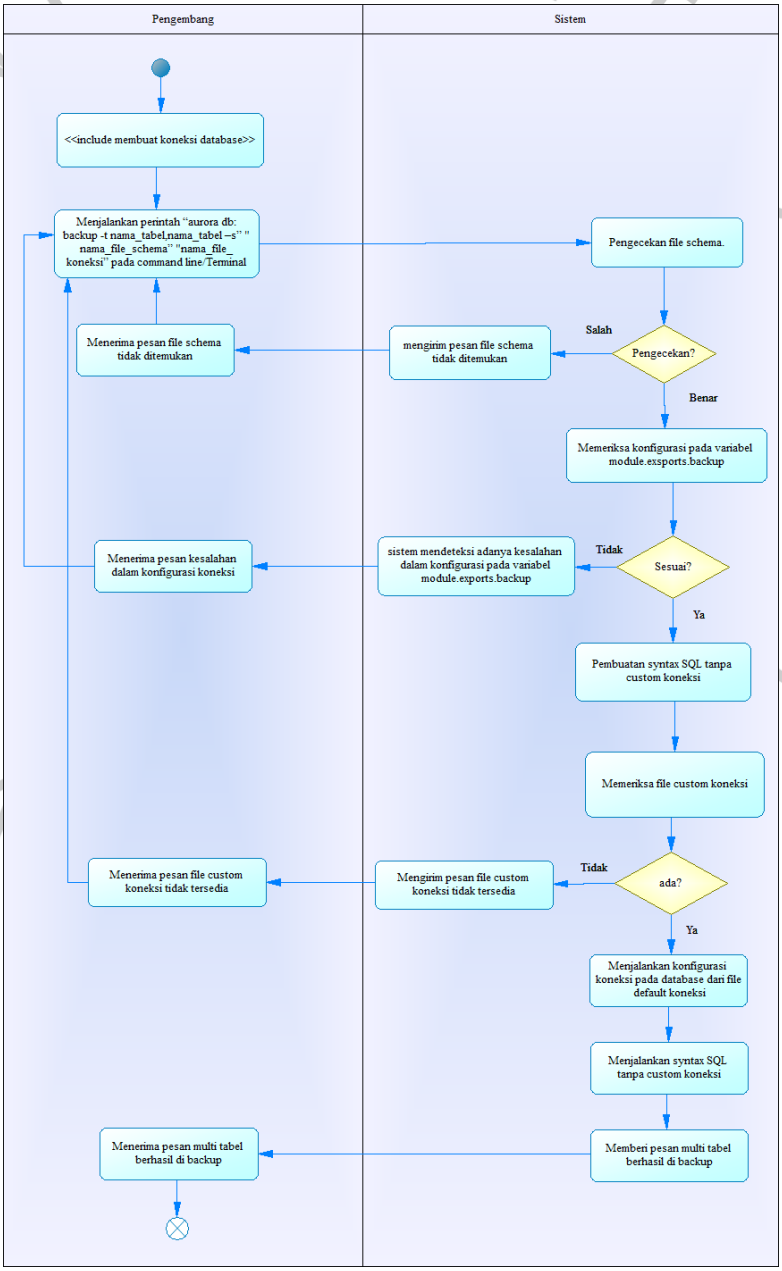
Gambar 4.9. *Activity Diagram* Menjalankan Perintah *Backup Multi Table*

Pada **Gambar 4.10** menjelaskan proses *activity diagram* dalam menjalankan *backup multi table* tanpa custom koneksi, untuk menjalankan perintah pengembang sudah melakukan *default* koneksi terlebih dahulu dengan menggunakan kofigurasi koneksi yang sudah tersedia pada *framework* aurora.JS. Dapat dilihat pada gambar di bawah ini.



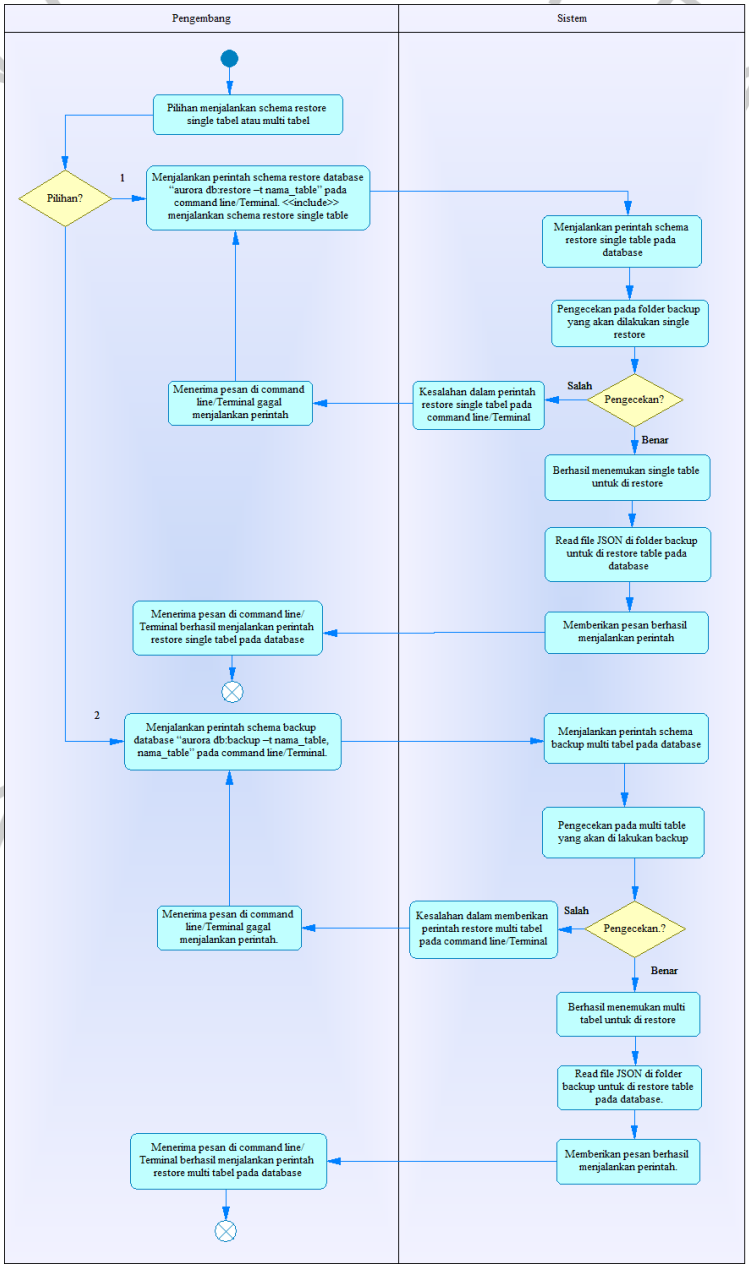
Gambar 4.10. *Activity Diagram* Membuat *Backup Multi Table* Tanpa *Custom* Koneksi

Pada **Gambar 4.11** proses *activity diagram* dalam membuat *backup multi table* dengan *custom* koneksi, untuk melakukan *custom* koneksi pengembang dapat melakukan *custom* koneksi terlebih dahulu dengan menambah penamaan saat melakukan konfigurasi koneksi, selanjutnya saat melakukan *backup multi table* pengembang dapat menambahkan nama koneksi agar *file* konfigurasi koneksi dapat menjalankan *backup file* yang sudah dilakukan oleh pengembang pada *framework* Aurora.JS



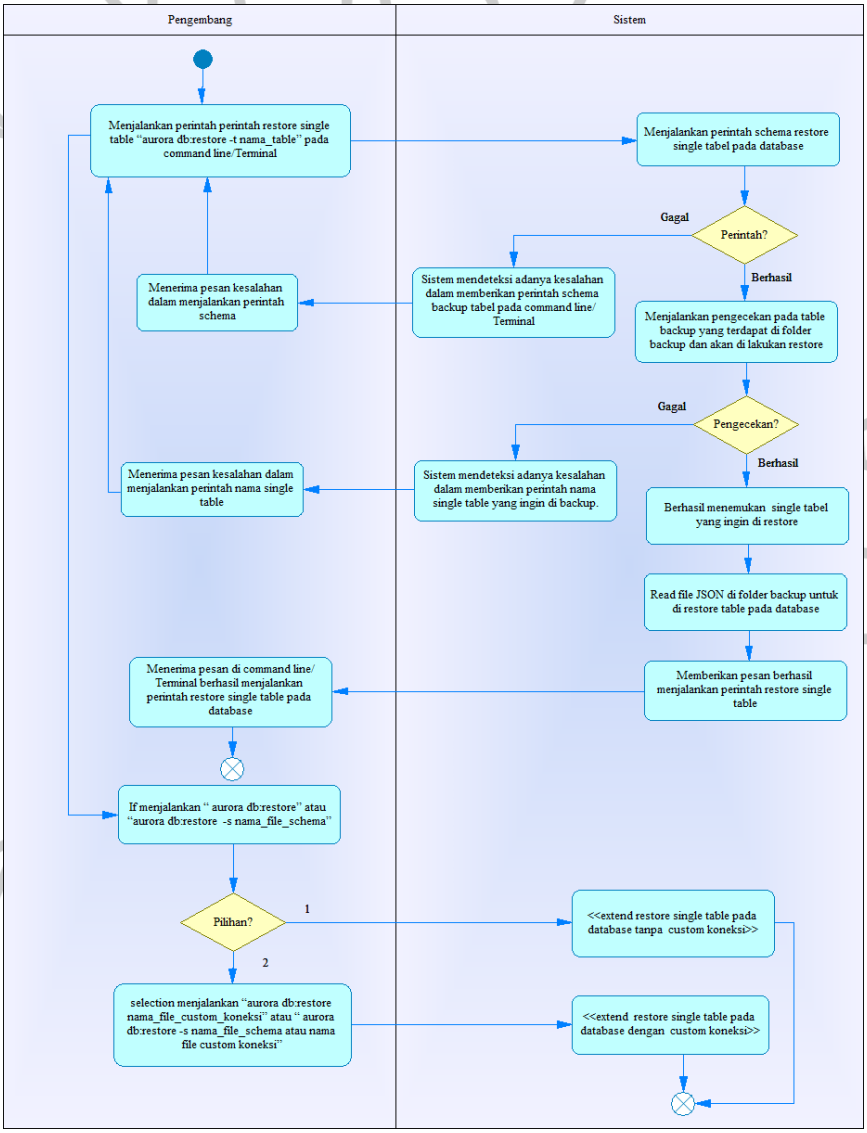
Gambar 4.11. *Activity Diagram* Membuat *Backup Multi Table* Dengan *Custom* Koneksi

Dapat dilihat pada **Gambar 4.12.** Merupakan alur sistem dari fitur *restore table* database pada *framework* Aurora.JS, pada fitur ini untuk mengembalikan data *table* yang ingin di *restore* harus sudah menjalankan perintah untuk melakukan *backup table* terlebih dahulu, jika sudah akan terjadinya penambahan file JSON di folder *backup*. Setelah itu pengembang dapat menjalankan perintah pada *command-line* (CMD) yaitu *restore table* pada database pada file yang sudah dilakukan *backup*.



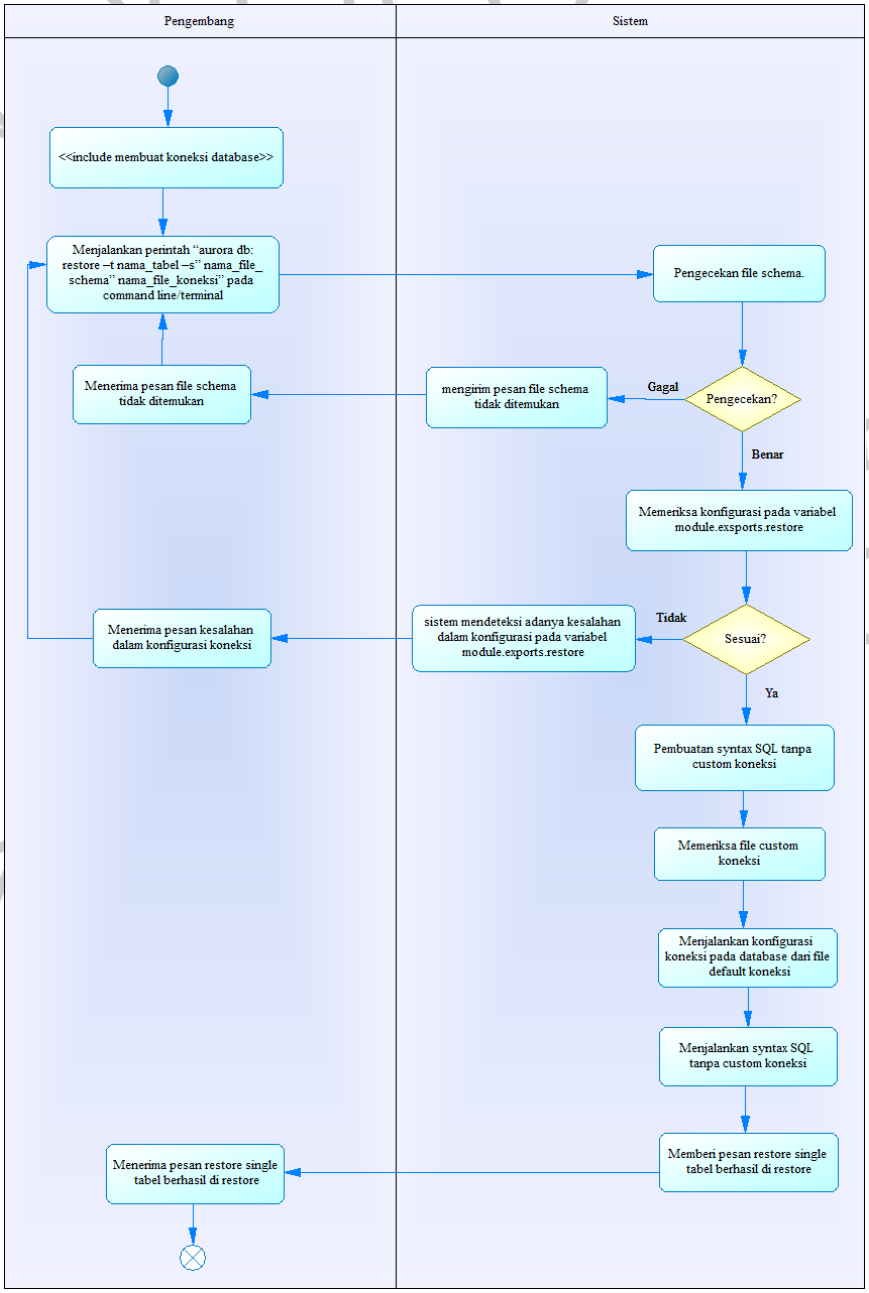
Gambar 4.12. *Activity Diagram* Menjalankan Perintah *Restore*

Pada **Gambar 4.13** menjelaskan *Activity Diagram* mengenai *restore single table*, pada tahap ini data *table* yang sudah di *backup* akan di *restore* ke dalam *table* pada database, untuk menjalankan *restore single table* sudah menjalankan perintah *backup* terlebih dahulu, selanjutnya pengembang dapat menjalankan perintah *restore single table*, berikut adalah penjelasan pada gambar di bawah ini.



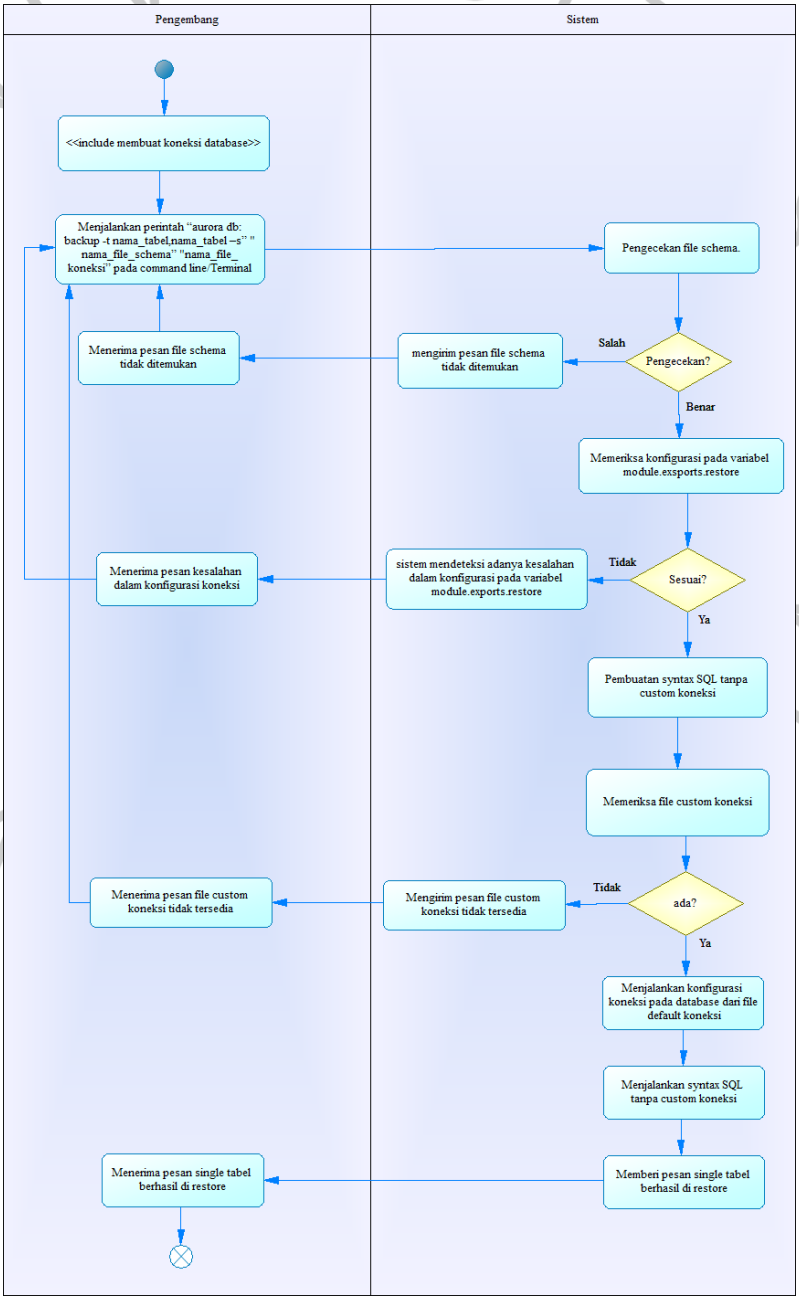
Gambar 4.13. *Activity Diagram* Menjalankan Perintah *Restore Single Table*

Pada **Gambar 4.14** tahapan ini pengembang dapat menentukan *restore single table* tanpa *custom* koneksi, dari perintah tersebut sistem akan memproses koneksi pada *framework* Aurora.JS ke database, dalam mengembalikan data *table* yang sudah di *backup* ke dalam *table* database dengan menjalankan konfigurasi *default* koneksi sesuai tanpa melakukan menambahkan penamaan perintah dalam menjalankan konfigurasi koneksi.



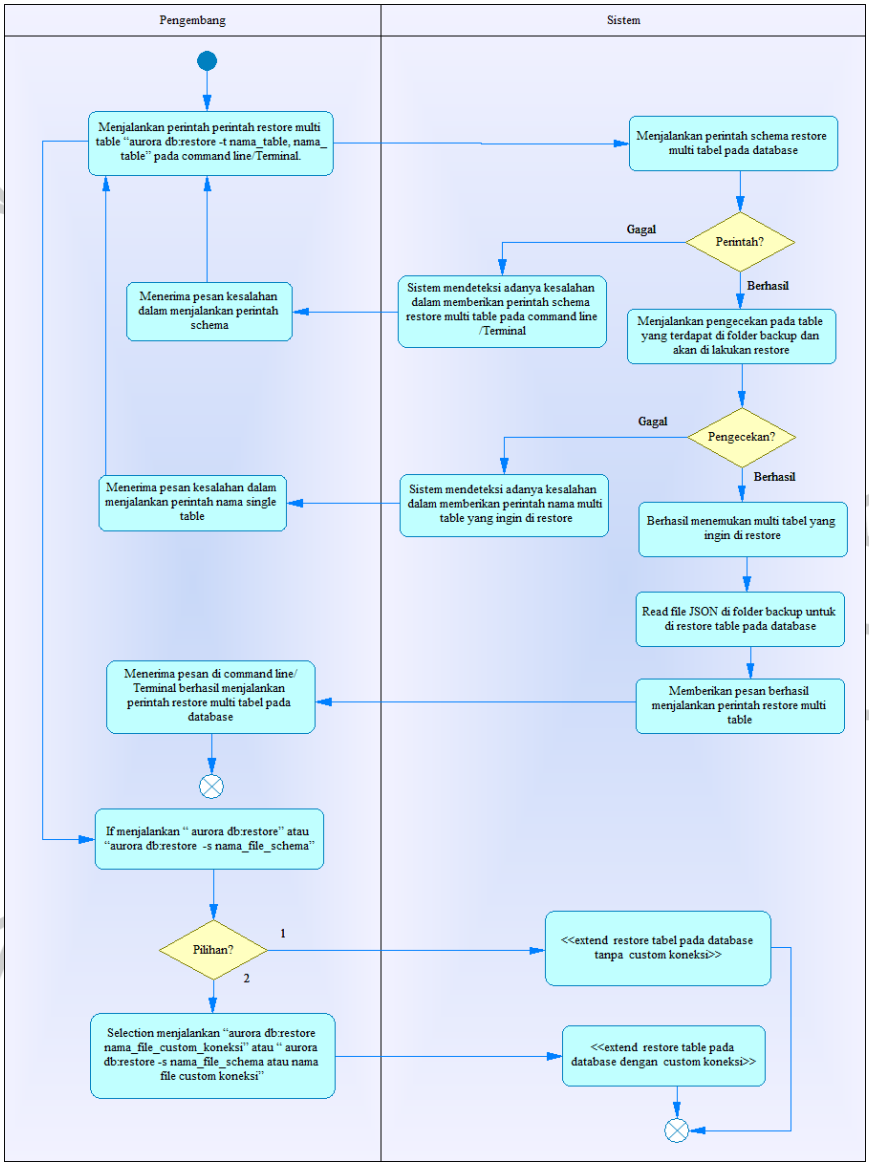
Gambar 4.14. *Activity Diagram Read Restore Single Table* Tanpa *Custom* Koneksi

Pada **Gambar 4.15** tahap ini pengembang dapat menentukan *restore single table* dengan *custom* koneksi, dari perintah tersebut sistem akan memproses koneksi ke database yang sudah di *custom* dengan menambahkan nama perintah *custom* koneksi pada *framework* Aurora.JS dari hal ini mempermudah konfigurasi *custom* koneksi pada pengembang dalam mengembalikan data *table* yang sudah di *backup* dan dikembalikan ke dalam *table* database.



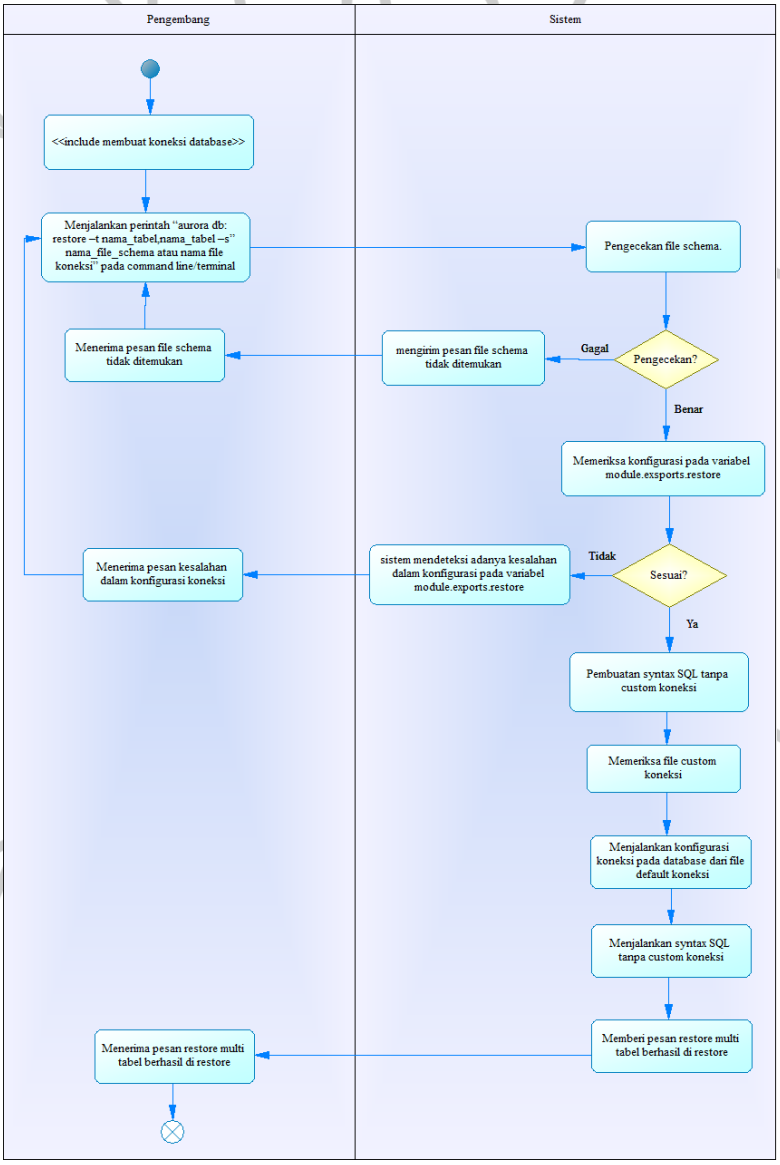
Gambar 4.15. *Activity Diagram Read restore single table* Dengan *Custom* Koneksi

Pada **Gambar 4.16** menjelaskan *Activity Diagram* mengenai *restore multi table*, pada tahap ini data *table* yang sudah di *backup* akan di *restore* ke dalam *table* pada database yang sudah ditentukan, untuk menjalankan *restore multi table* harus sudah menjalankan perintah *backup* terlebih dahulu, selanjutnya pengembang dapat menjalankan perintah *restore multi table*.



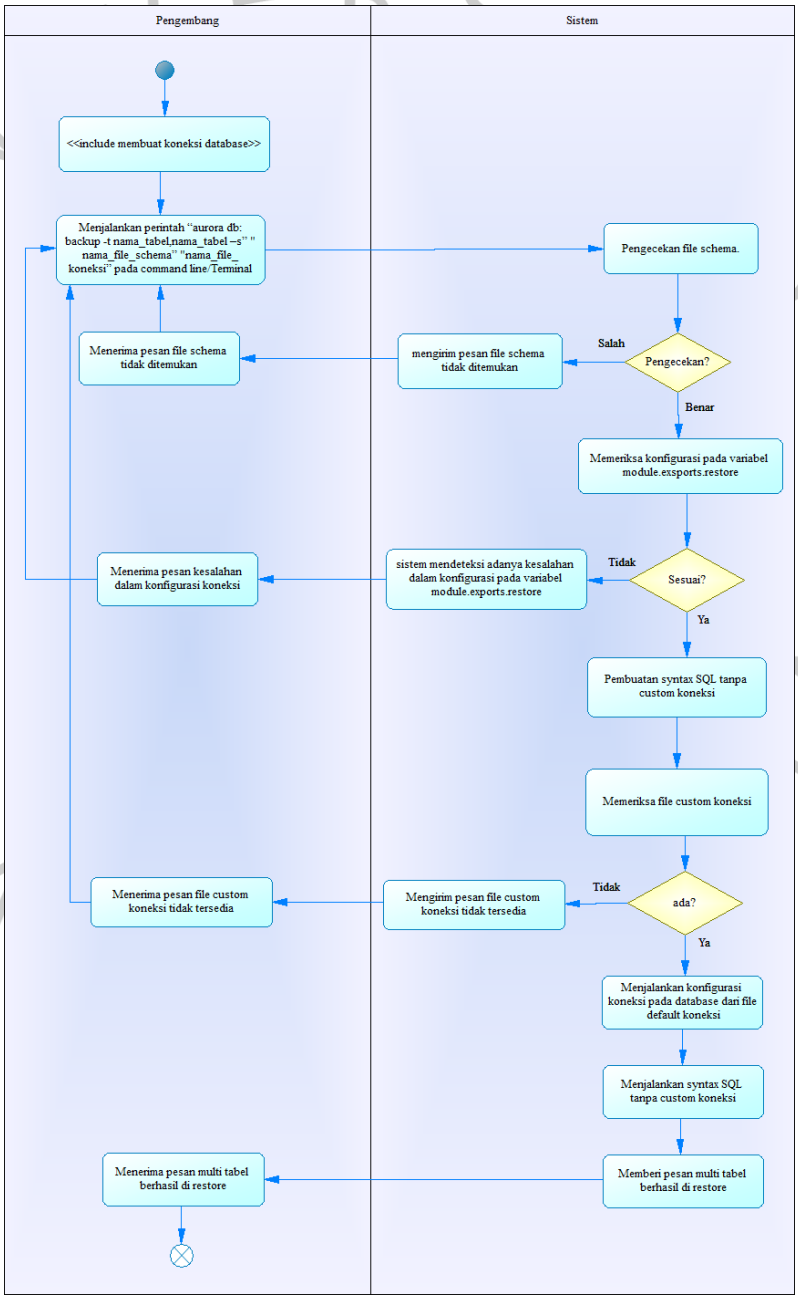
Gambar 4.16. *Activity Diagram* Menjalankan Perintah *Restore Multi Table*

Pada **Gambar 4.17** tahapan ini pengembang dapat menentukan *restore multi table* tanpa *custom* koneksi, dari perintah tersebut sistem akan memproses koneksi dari framework Aurora.JS ke database, dalam mengembalikan data *table* yang sudah di *backup* ke dalam *table*, dalam menjalankan konfigurasi *default* koneksi dengan sesuai tanpa melakukan menambahkan penamaan perintah dalam menjalankan konfigurasi koneksi.



Gambar 4.17. *Activity Diagram Read Restore Multi Table* Tanpa *Custom* Koneksi

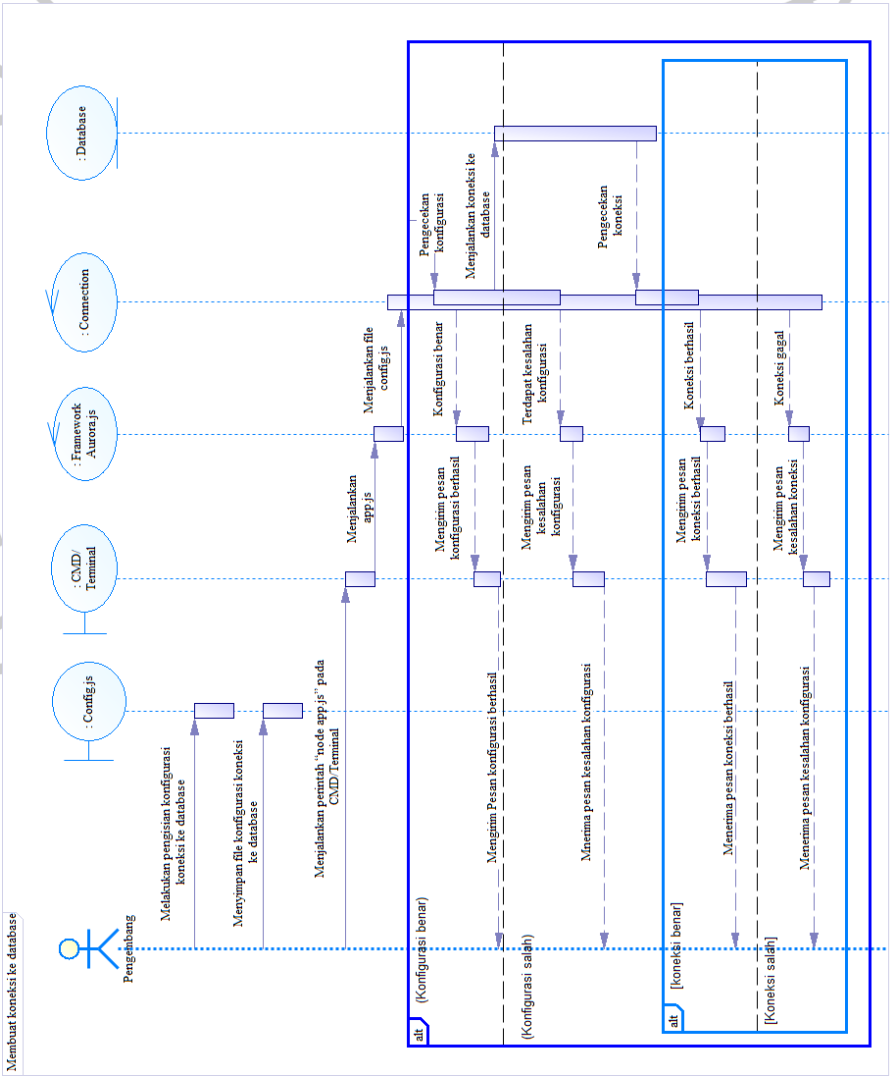
Pada **Gambar 4.18** tahap ini pengembang dapat menentukan *restore multi table* dengan *custom* koneksi, dari perintah tersebut sistem akan memproses koneksi ke database yang sudah di *custom* dengan menambahkan nama perintah *custom* koneksi pada *framework* Aurora.JS dari hal ini mempermudah konfigurasi *custom* koneksi pada pengembang dalam mengembalikan data *table* yang sudah di *backup* dan dikembalikan ke dalam *table* database.



Gambar 4.18. *Activity* Diagram *Read Restore Multi Table* Dengan *Custom* Koneksi

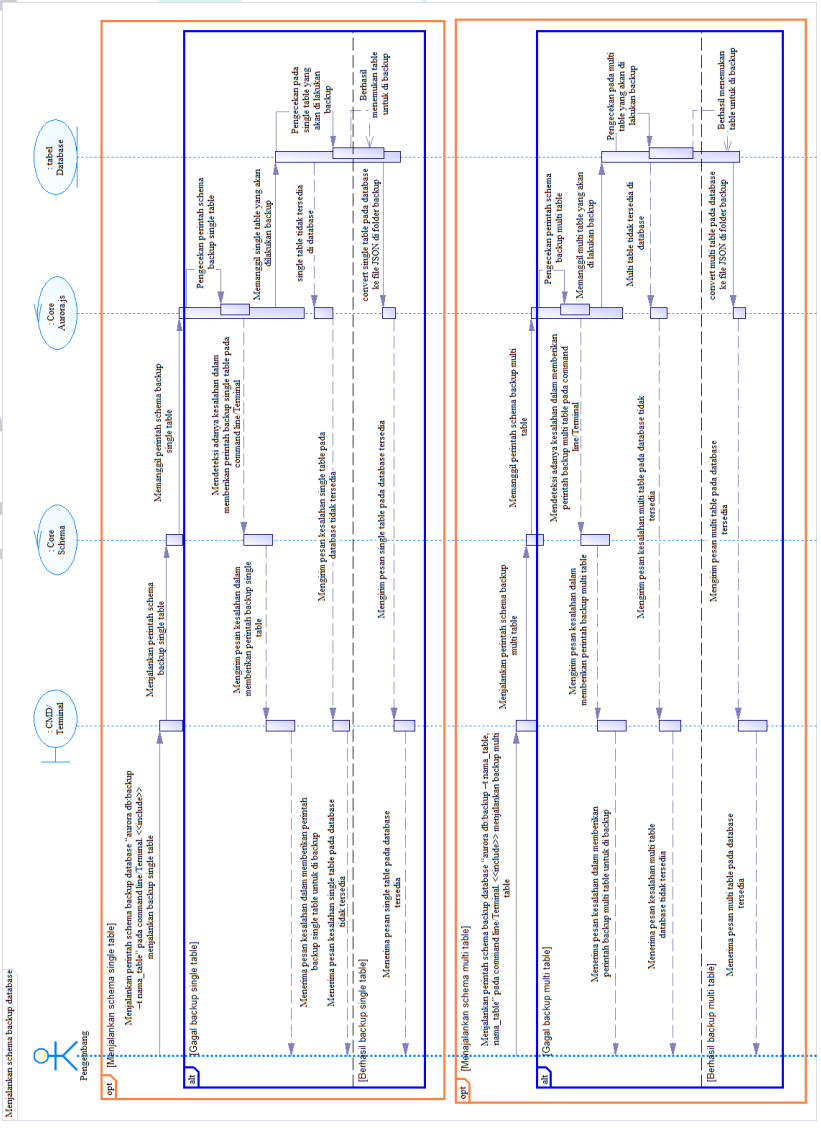
4.2.4 *Sequence* Diagram

Penggunaan *sequence* diagram dimaksudkan untuk menjelaskan secara detail proses interaksi antara pengembang dan sistem. Detail interaksi tersebut dapat dilihat pada partisi sistem yang terdiri dari modul *backup* dan *restore table* database pada *framework* Aurora.JS, pada **Gambar 4.19.** Merupakan *Sequence Diagram* dalam membuat koneksi database pada *framework* Aurora.JS.

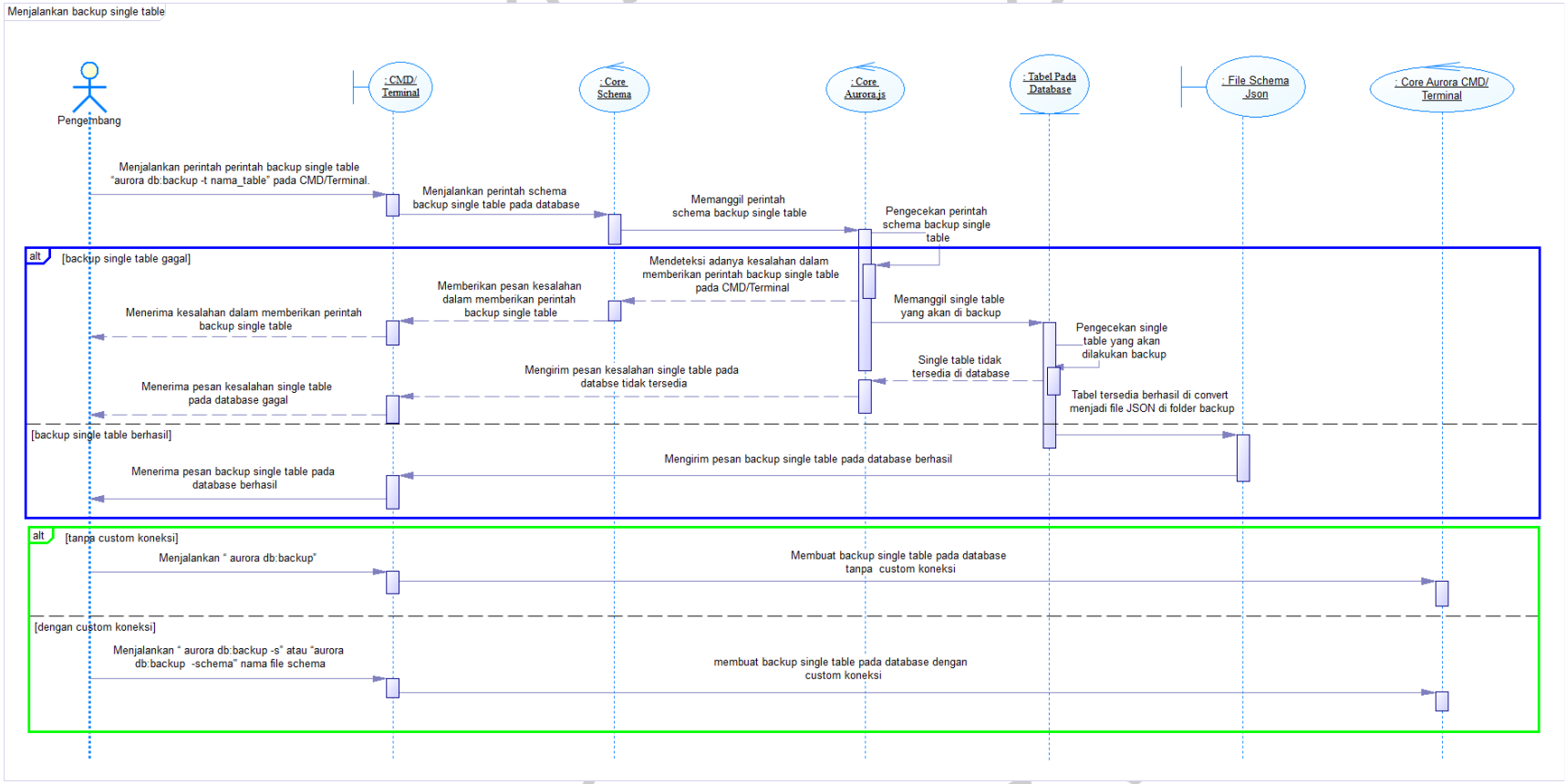


Gambar 4.19. *Sequence Diagram* Membuat Koneksi Database

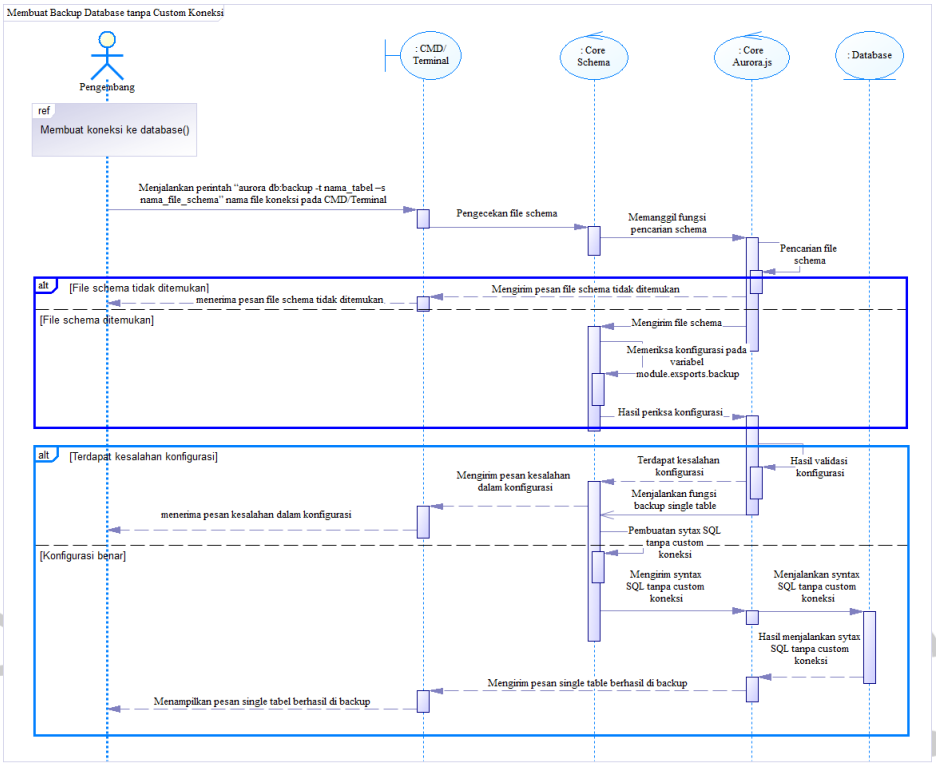
Pada **Gambar 4.20 – 4.26.** Merupakan penjelasan dalam menjalankan modul pada fitur *backup table* pada database. Tahapan pada gambar tersebut dimulai dengan bagaimana alur sistem dan pengembang dalam menjalankan perintah *backup table*, *single table* dan *multi table*. Selanjutnya dari hasil rancangan tersebut pengembang juga dapat melakukan *custom* koneksi dan *default* koneksi dalam menjalankan fitur *backup table*. Setelah pengembang menjalankan perintah *backup table*, sistem akan melakukan proses penyimpanan data ke dalam format JSON yang akan disimpan pada *framework* Aurora.JS yang sudah ditentukan penempatan folder agar dapat terstruktur dan mudah dipahami oleh pengguna dan pengembang.



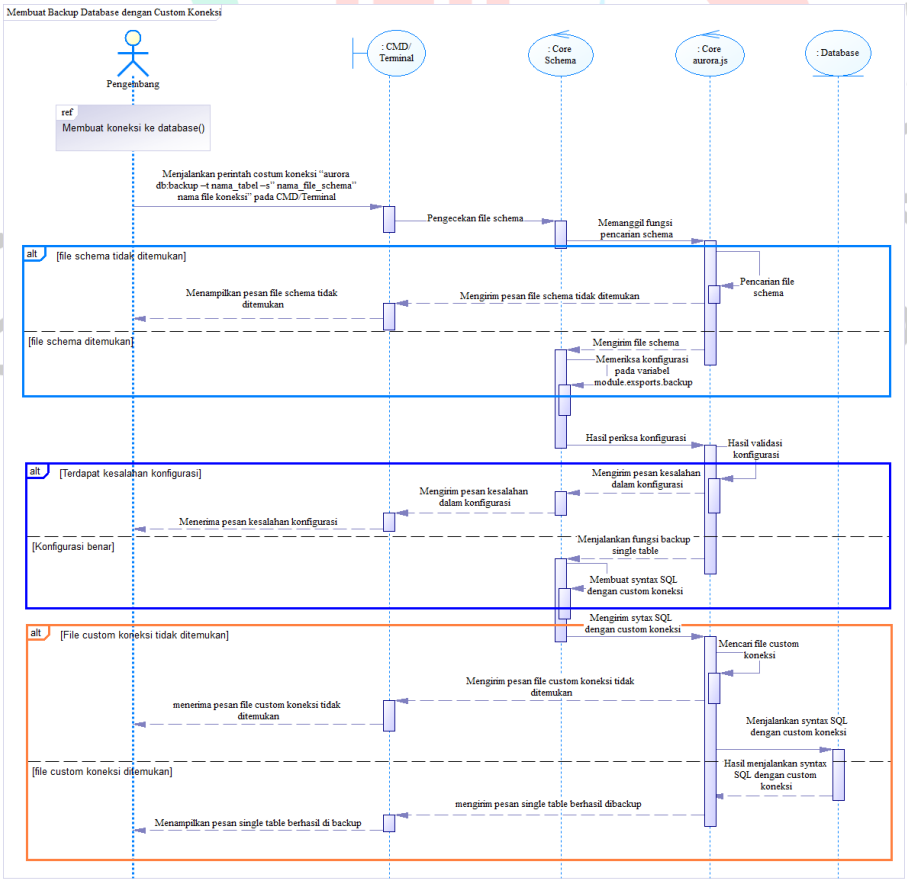
Gambar 4.20. *Sequence Diagram* Menjalankan Perintah *Backup*



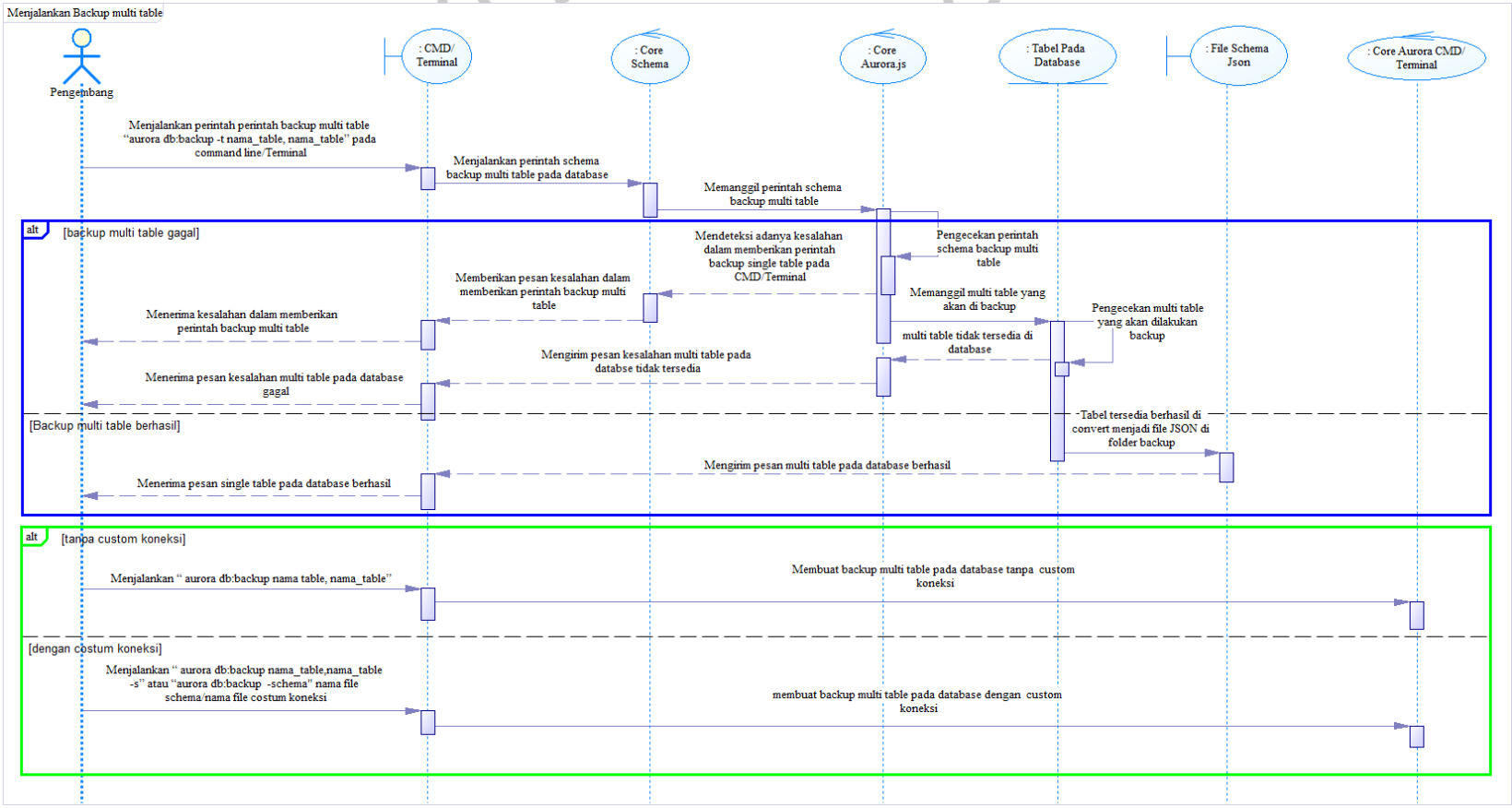
Gambar 4.21. *Sequence Diagram* Menjalankan Perintah *Backup Single Table*



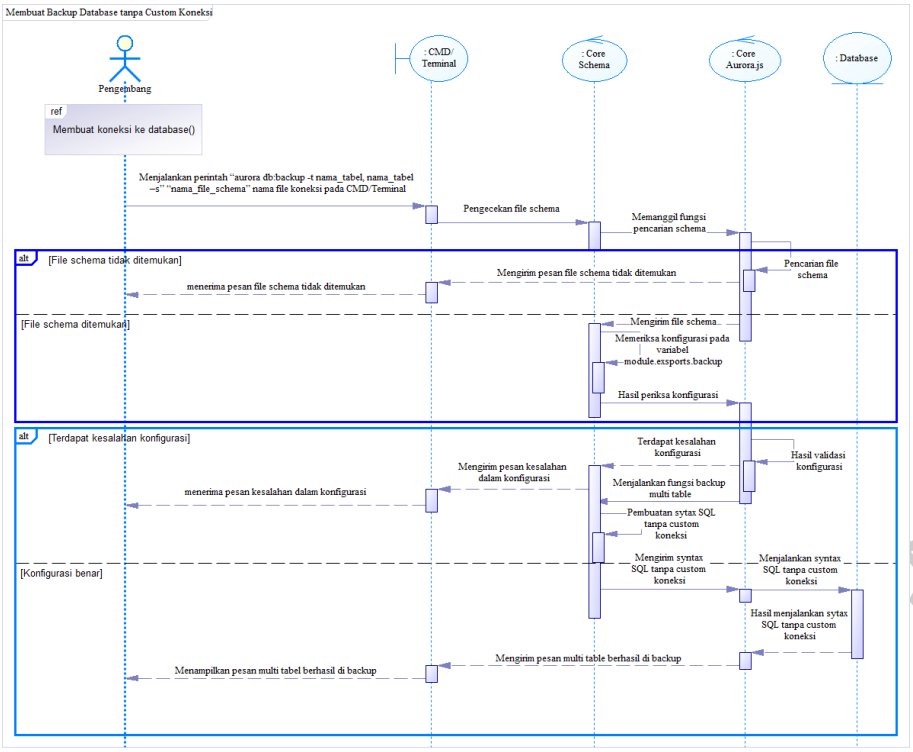
Gambar 4.22. *Sequence Diagram* Membuat *Backup single table* Tanpa *Custom* Koneksi



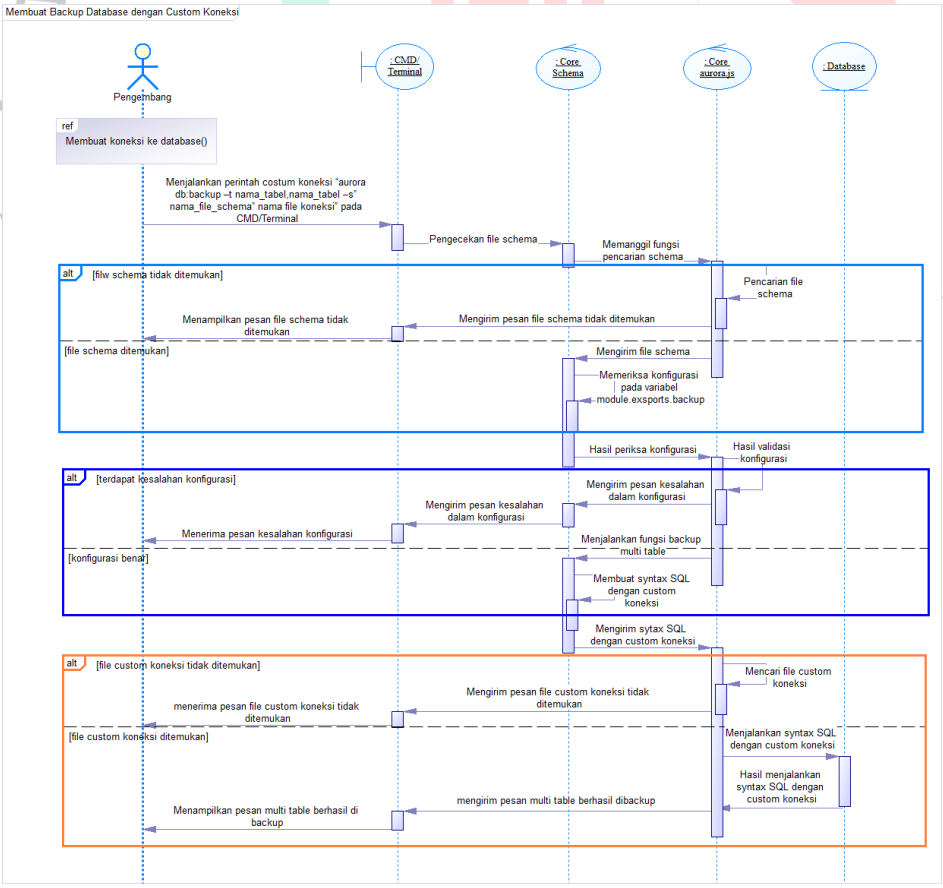
Gambar 4.23. *Sequence Diagram* Membuat *Backup Single Table* Dengan *Custom* Koneksi



Gambar 4.24. *Sequence Diagram* Menjalankan Perintah *Backup Multi Table*



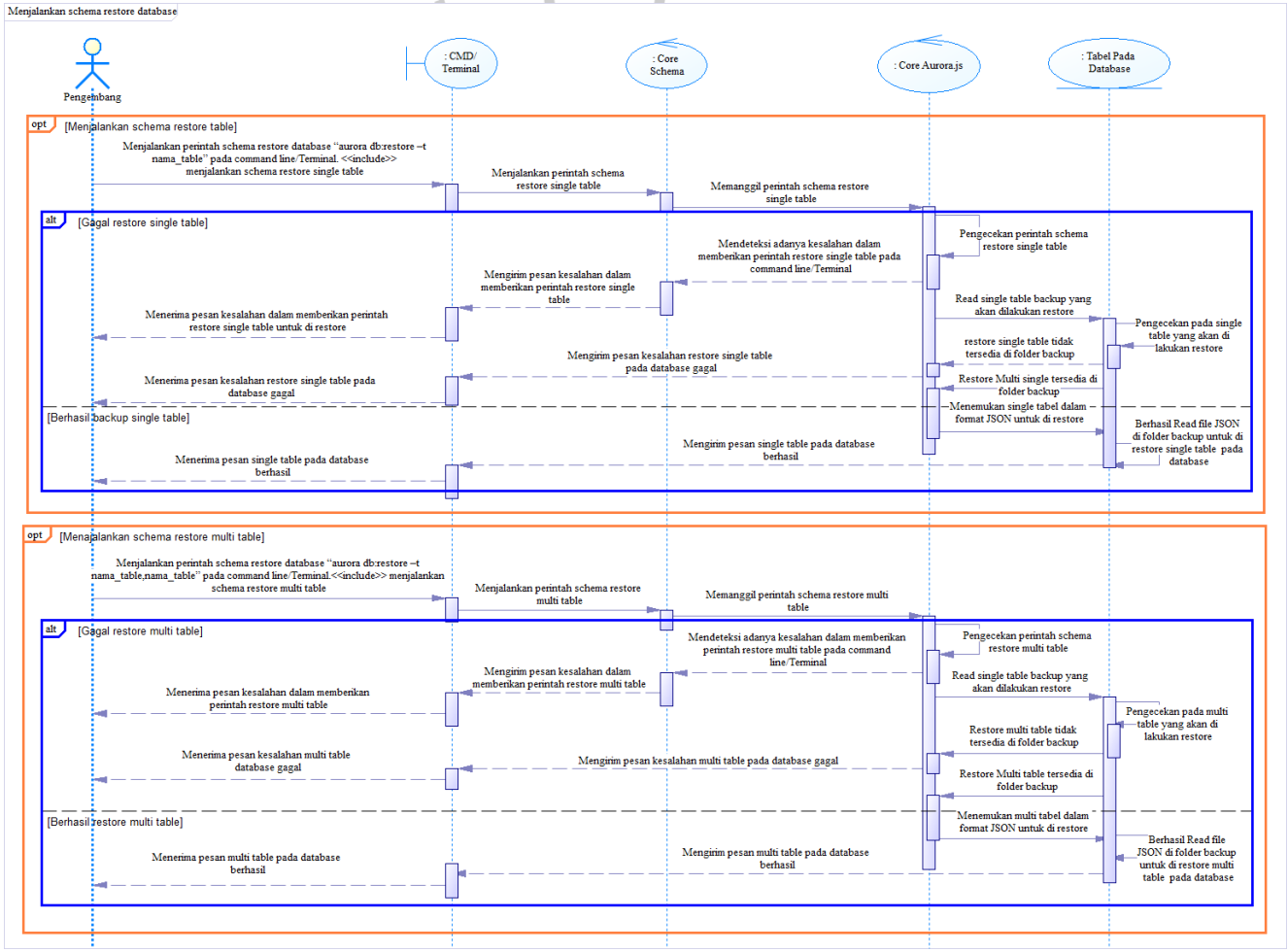
Gambar 4.25. *Sequence Diagram* Membuat *Backup Multi Table* Tanpa *Custom* Koneksi



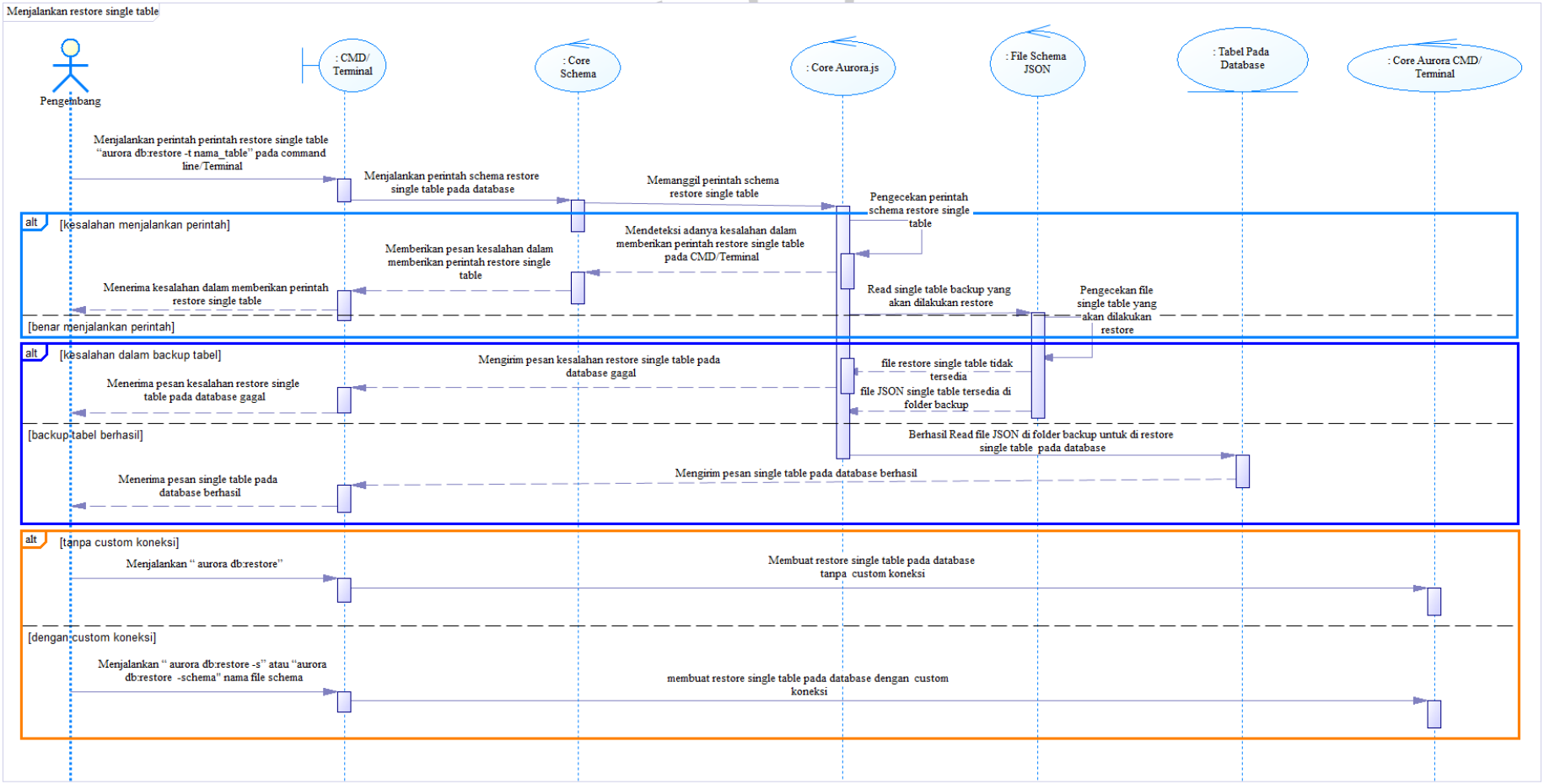
Gambar 4.26. *Sequence Diagram* Membuat *Backup Multi Table* Dengan *Custom* Koneksi

Berikut ini penjelasan mengenai rancangan *Sequence Diagram* dalam menjalankan perintah *restore table*, dalam rancangan *Sequence Diagram* menjelaskan bagaimana cara *restore* data *table* yang sudah dilakukan *backup file* terlebih dahulu setelah menjalankan perintah *backup*, terdapat *file* JSON yang sudah dilakukan *backup* terlebih dahulu dan sudah terkoneksi ke database, dari hal tersebut *file* yang sudah di *backup* akan di *convert* dalam *file* JSON dan data tersebut akan tersimpan pada folder *backup* pada *framework* Aurora.JS, jika sudah pengembang dapat menjalankan perintah *restore table* pada *command line.*

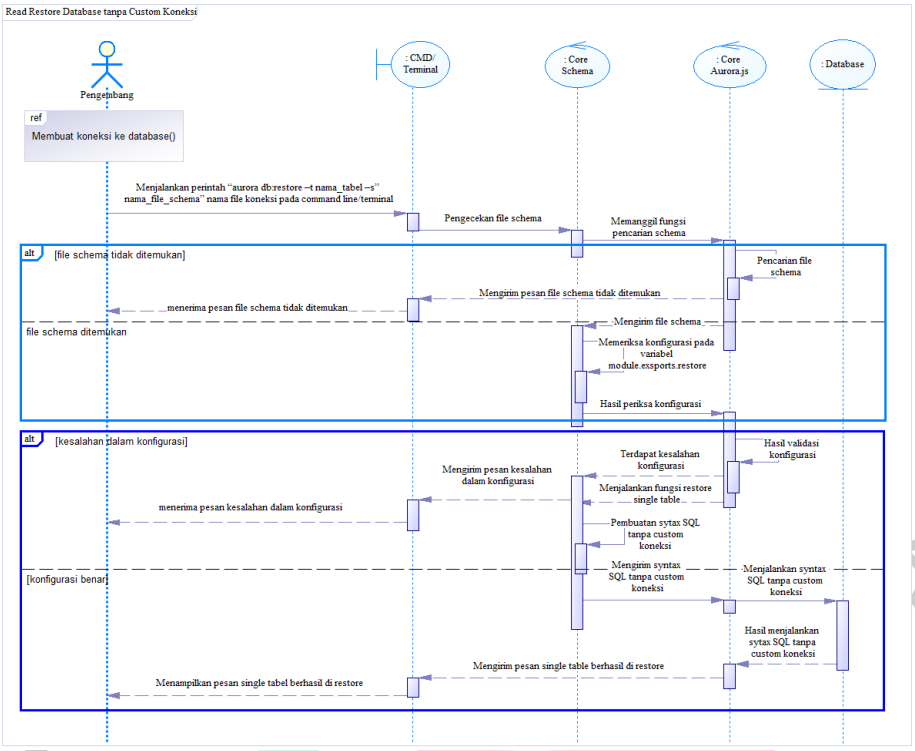
fitur *restore* akan membaca *file* tersebut dan mengembalikan seluruh data ke dalam database pengembang, tentu data yang sudah di *backup* akan masuk ke dalam database pada *table* yang sudah ditentukan untuk di *restore*. Selanjutnya dalam menjalankan Fitur *restore table* pada database pengembang dapat melakukan pilihan dengan menjalankan perintah *restore single table* dengan melakukan *restore* hanya satu *table* saja dan menjalankan perintah *multi table* dengan beberapa *table* yang diinginkan untuk di *restore* sesuai dengan apa yang pengembang atau pengguna inginkan dalam menjalankan perintah *restore*, dalam fitur *restore table* pada database pengembang dan pengguna juga dapat melakukan *custom* koneksi atau *default* koneksi dengan menambah penamaan dalam menjalankan perintah. Dari hasil perintah yang dijalankan maka sistem akan menyimpan data *restore table* ke dalam database yang sudah ditentukan, hal ini merupakan rancangan alur sistem dan pengembang dalam menjalankan sistem melalui analisis *Sequence Diagram* dari fitur *restore table* berikut ini rancangan Sequence diagram pada **Gambar 4.27 – 4.33**.



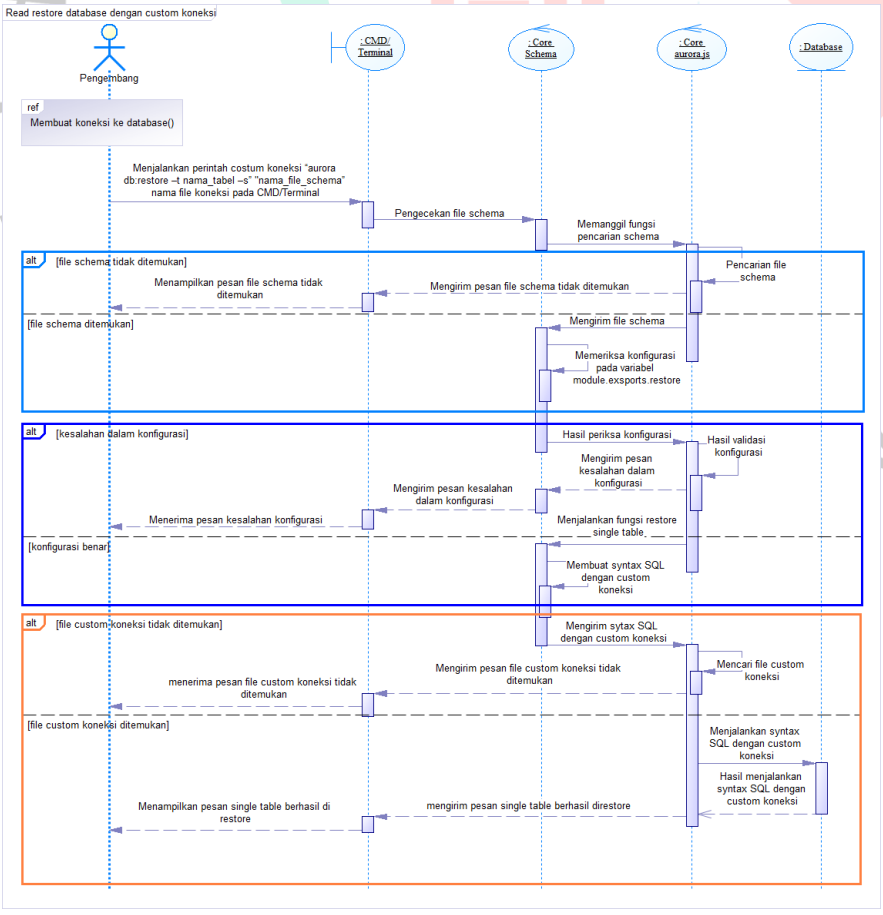
Gambar 4.27. *Sequence Diagram* Menjalankan Perintah *Restore*



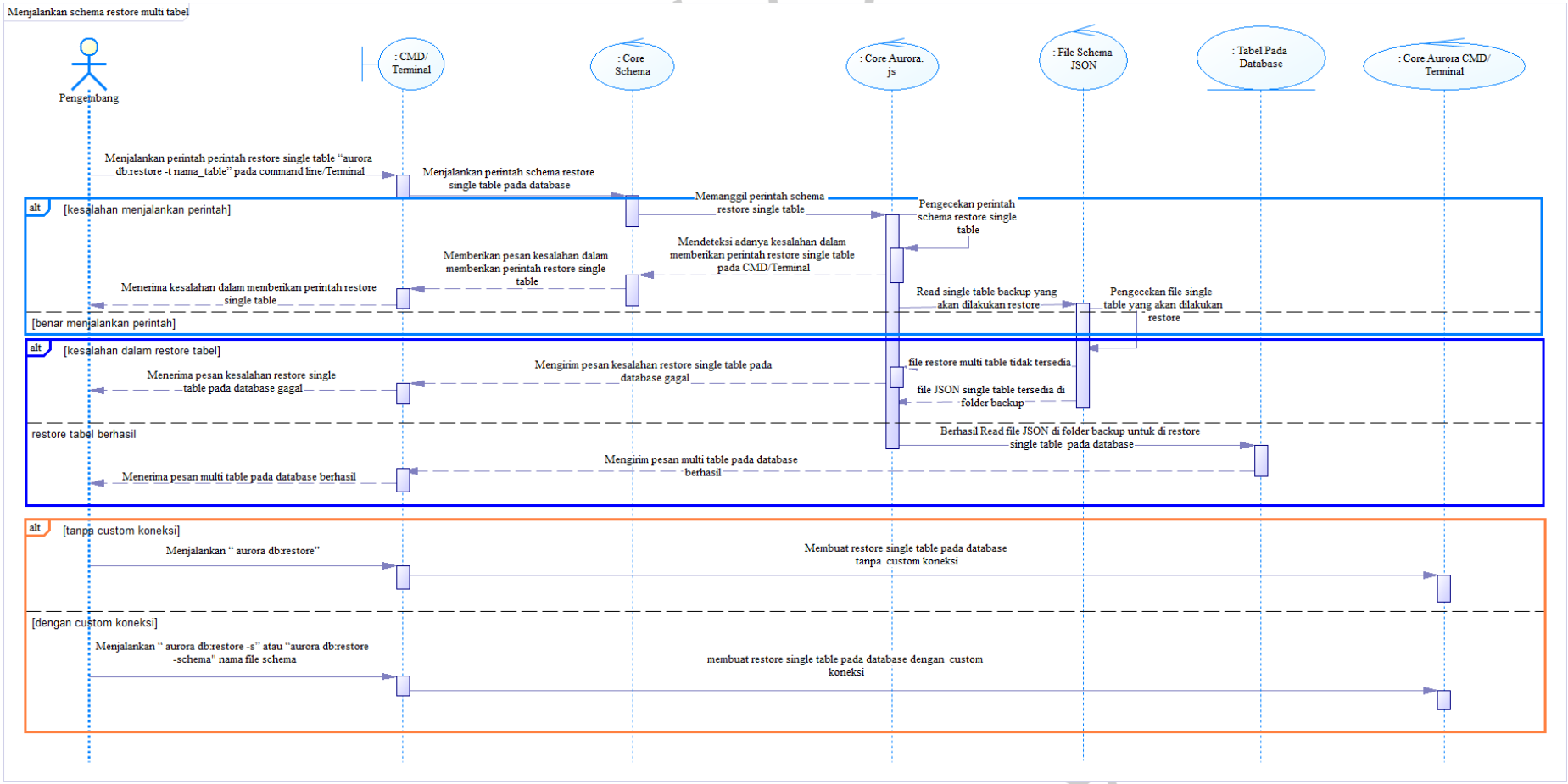
Gambar 4.28. *Sequence Diagram* Menjalankan Perintah *Restore Single Table*



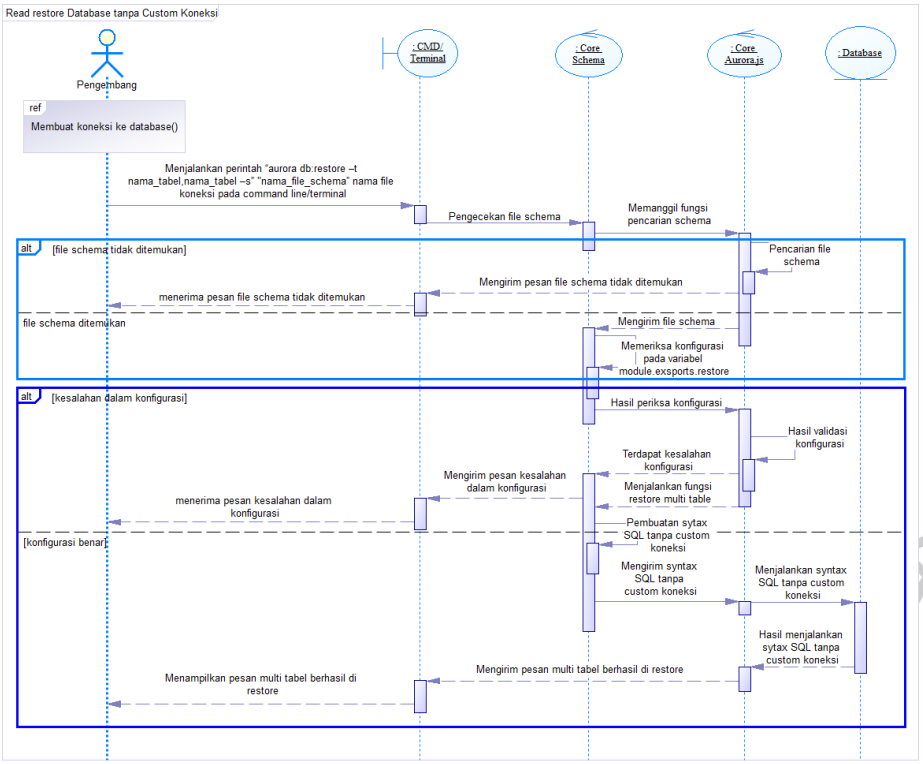
Gambar 4.29. *Sequence Diagram Read Restore Single Table* Tanpa *Custom* Koneksi



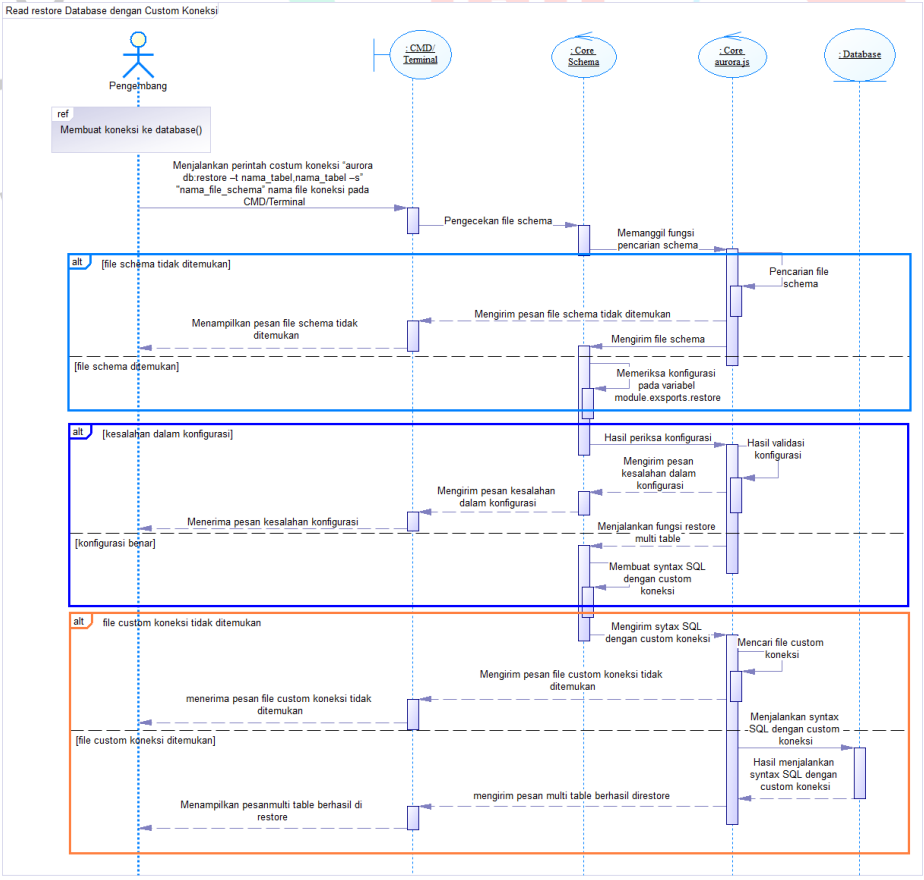
Gambar 4.30. *Sequence Diagram Read Restore Single Table* Dengan *Custom* Koneksi



Gambar 4.31. *Sequence Diagram* Menjalankan Perintah *Restore Multi Table*



Gambar 4.32. *Sequence Diagram Read Restore multi table* Tanpa *Custom* Koneksi



Gambar 4.33. *Sequence Diagram Read Restore Multi Table* Dengan *Custom* Koneksi

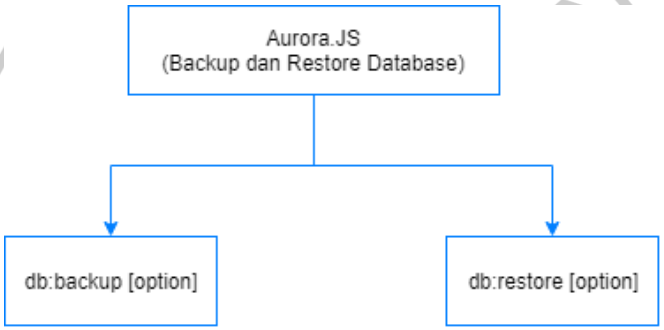
4.3 Perancangan Antar Muka Pengguna

Dari rancangan penulis yang sudah dikembangkan mengenai fitur *backup* dan *restore* database, peneliti menjelaskan interaksi dari rancangan antara pengembang dengan *framework* Aurora.JS, fungsi tersebut akan dijalankan dengan memberikan perintah melalui *Command line* atau terminal, untuk penggunaan perintah yang penulis rancang modul *backup* dan *restore table* pada database akan dijelaskan rancangan interaksi di bawah ini :

4.3.1 Struktur Menu

Fitur *backup* dan *restore table* database pada *Framework* Aurora.JS memiliki rancangan struktur menu pada gambar **4.33** berikut ini penjelasan perintah yang dapat digunakan pada *Command line* atau terminal sebagai berikut:

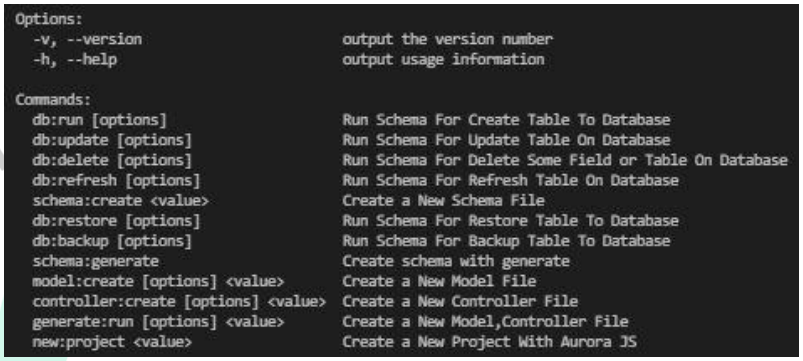
1. Menu pada fitur db:*backup* [option] digunakan untuk *backup table* pada database yang nantinya dapat digunakan sewaktu-waktu jika pengembang membutuhkan data *table* tersebut.
2. Menu pada fitur db:*restore* [option] digunakan untuk *restore table* pada database, yang nantinya data *table* yang sudah dilakukan *backup* di f*ramework* Aurora.JS, selanjutnya data tersebut akan dikembalikan ke dalam database sesuai dengan konfigurasi koneksi yang digunakan.



Gambar 4.34. Struktur Menu Aurora.JS Modul *Backup* dan *Restore Table* Pada Database

4.3.2 Rancangan Layout atau Tampilan

Pada *framework* Aurora.JS memiliki interaksi pada pengembang dengan menggunakan rancangan layout melalui *command line* (CMD) atau terminal. Pengembang dapat melakukan *input text* dengan menjalankan perintah yang tersedia, dapat dilihat pada **Gambar 4.35.**



Gambar 4.35. Menu Dalam Menjalankan Perintah Pada Aurora.JS

4.3.3 Perancangan Masukan (Input)

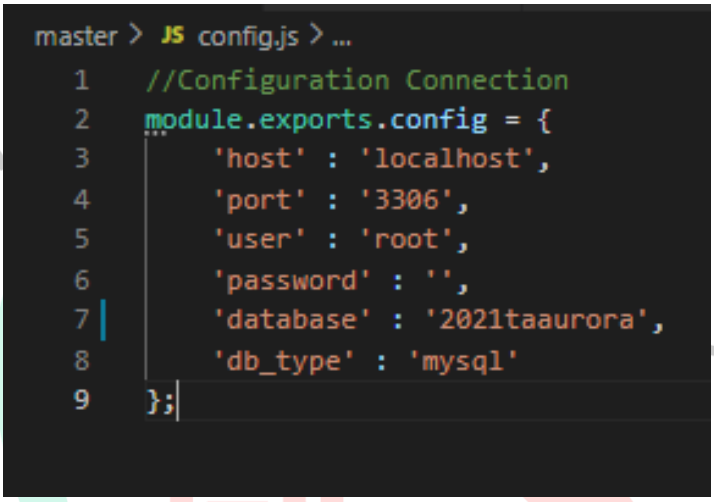
Rancangan fitur pada modul *backup* dan *restore table* pada database yang dirancang di *framework* Aurora.JS dapat digunakan pengembang dengan menjalankan perintah yang tersedia, berikut ini perintah yang dapat digunakan untuk menjalankan fitur *backup* dan *restore table pada* database :

Tabel 4.16. Menjalankan Perintah pada *Framework* Aurora.JS

No	Perintah Command line (CMD) atau terminal		Keterangan
1	db:backup		Menjalankan backup tabel pada database
2	Db:backup -t		
2	db:restore		Menjalankan restore tabel dalam mengembalikan data yang sudah di *backup* pada framework aurora.js ke dalam database
3	[Option]	-s	Menjalankan *file* schema yang diinginkan
4		<value>	Untuk mengisi nama *file custom* koneksi/ pemberian nama file
5	Node app.js		Perintah untuk menjalankan koneksi

Hal yang dilakukan pada pengembang dalam menjalankan fitur *backup* dan *restore table* pada database yaitu dengan melakukan koneksi ke database terlebih dahulu, koneksi dapat

dilakukan dengan cara memasukan data dapat dilihat pada **Gambar 4.36** yaitu konfigurasi koneksi tanpa *custom* koneksi, pada variabel di dalam *file* schema Config.JS untuk menjalankan koneksi tersebut hanya dapat digunakan oleh MySQL saja.



Gambar 4.36. Konfigurasi Koneksi Tanpa *Custom* Koneksi Pada Database

Dalam menjalankan *file* pada Config.JS diperlukan perintah untuk *run file* tersebut, yaitu dengan cara menjalankan perintah pada *command line* (CMD) atau terminal seperti perintah di bawah ini menjalankan perintah koneksi.

```
C:\xampp\htdocs\aurora\master>node app.js
Aurora serve on port 3000
```

Untuk menjalankan perintah koneksi ke database terdapat juga untuk *file custom* koneksi yang sangat berguna bagi pengembang agar dapat lebih mudah dalam melakukan konfigurasi dan pekerjaan secara tim, untuk menjalankan konfigurasi koneksi dengan *custom* koneksi yang dirancang pengembang dapat dilihat pada **gambar 4.37** *custom file* konfigurasi koneksi.

Gambar 4.37. Konfigurasi Koneksi Dengan *Custom* Koneksi Pada Database

Dalam menjalankan perintah *custom file* konfigurasi koneksi dapat dilakukan dengan menambah parameter di akhir perintah yaitu dengan memberi nama *custom file* koneksi yang diinginkan pengguna pada *framework* Aurora.JS, dapat dilihat di bawah ini merupakan perintah menjalankan *custom file* konfigurasi koneksi.

```
C:\xampp\htdocs\aurora\master>node app.js dhoni
Aurora serve on port 3000
```

Setelah menjalankan koneksi dari *framework* Aurora.JS ke database pengembang dapat menjalankan fitur pada modul *backup* dan *restore table* pada database, dengan syarat memiliki *table* pada database. Dalam menjalankan perintah *backup* dapat dilihat perintah di bawah ini yaitu Menjalankan perintah *backup table* pada database.

```
C:\xampp\htdocs\aurora\master>aurora db:backup
```

Selanjutnya untuk melakukan *backup single table* pada database di *framework* Aurora.JS, untuk menjalankan fitur *backup single table* pengembang dapat memberikan perintah di *command-line* seperti di bawah ini :

```
C:\xampp\htdocs\aurora\master>aurora
db:backup –t datakaryawan
```

Pada perintah di bawah ini merupakan perintah untuk menjalankan *backup single table* dengan *custom* koneksi pada *framework* Aurora.JS, untuk menjalankan perintah *backup multi table* dengan *custom* koneksi dapat memberikan perintah di *command line* seperti:

```
C:\xampp\htdocs\aurora\master>aurora
db:backup –t datakaryawan nama_custom_koneksi.
config.js
```

Dapat dilihat perintah di bawah ini merupakan perintah untuk menjalankan backup *multi table* pada database di *framework* aurora.JS, untuk menjalankan perintah *backup multi table* pengembang dapat memberikan perintah di *command-line* seperti di bawah ini :

```
C:\xampp\htdocs\aurora\master>aurora
db:backup –t datakaryawan,datasiswa
```

Pada perintah di bawah ini merupakan perintah untuk menjalankan *backup multi table* dengan *custom* koneksi pada framework Aurora.JS, untuk menjalankan perintah *backup multi table* dengan *custom* koneksi dapat memberikan perintah di *command line* seperti:

```
C:\xampp\htdocs\aurora\master>aurora
db:backup –t datakaryawan, datasiswa
nama_custom_koneksi.config.js
```

Setelah menjalankan *fitur backup* pada database, pengembang dapat melakukan *restore table* ke dalam database dengan syarat sudah menjalankan perintah *backup table* yang sudah

dilakukan oleh pengembang di *framework* Aurora.JS terlebih dahulu, hal ini sebagai syarat agar *file backup* dapat terbaca dan dapat dikembalikan kedalam database, dapat dilihat pada perintah di bawah ini Menjalankan perintah *restore table* pada database.

```
C:\xampp\htdocs\aurora\master>aurora
db:restore
```

Selanjutnya untuk menjalankan *restore single table* pada database pengembang dapat menjalankan perintah dengan menulis nama *table* yang sudah dilakukan *backup* seperti ini

```
C:\xampp\htdocs\aurora\master>aurora
db:restore –t datasiswa
```

Pada perintah di bawah ini merupakan perintah untuk menjalankan *restore single table* dengan *custom* koneksi pada *framework* Aurora.JS, untuk menjalankan perintah *restore single table* dengan *custom* koneksi dapat memberikan perintah di *command line* seperti:

```
C:\xampp\htdocs\aurora\master>aurora
db:restore –t datasiswa nama_custom_koneksi.
Config.js
```

Untuk menjalankan *restore multi table* pada database pengembang dapat menjalankan perintah dengan menambah *input* nama *table* yang sudah dilakukan *backup* untuk di *restore* ke dalam database pada pengembang seperti di bawah ini :

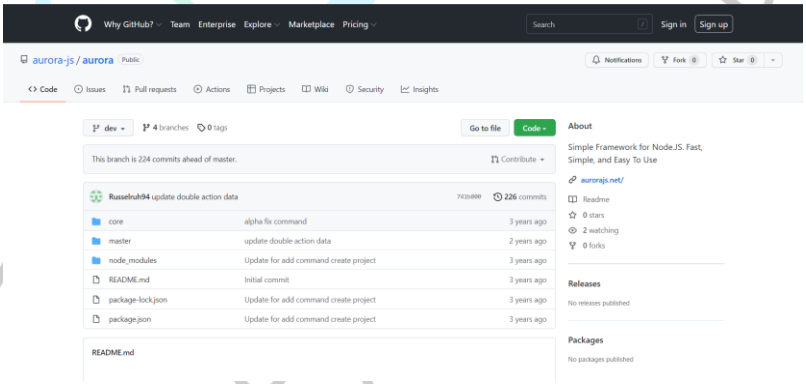
```
C:\xampp\htdocs\aurora\master>aurora
db:restore –t datasiswa,datakaryawan
```

Pada perintah di bawah ini merupakan perintah untuk

menjalankan *restore multi table* dengan *custom* koneksi pada *framework* Aurora.JS, untuk menjalankan perintah *restore multi table* dengan *custom* koneksi dapat memberikan perintah di *command line* seperti:

```
C:\xampp\htdocs\aurora\master>aurora db:restore -t dataksiswa, datakaryawan nama_custom_koneksi.config.js
```

Selanjutnya untuk berbagi data yang sudah tersimpan dalam bentuk JSON dapat dilakukan *apload* ke dalam github dengan begitu data yang sudah dalam format JSON dapat diolah dengan tim tanpa memberikan data *master* ke pihak ke 2 ataupun ke 3 karna data sudah tersimpan di dalam github, sehingga tim dapat melakukan *update* github pada *file* project yang dikerjakan pengguna dan pengembang untuk mengetahui data *table* yang diinginkan oleh tim untuk melakukan *backup* dan *restore* table pada database. Dari hal ini memberikan dampak dalam menyelesaikan pekerjaan secara cepat, dapat dilihat pada **Gambar 4.38.**



Gambar 4.38. *Sharing* Data Project Dengan Github

4.3.4 Perancangan Keluaran (*Output*)

Dari hasil proses *input* yang sudah dijelaskan dan dijalankan oleh penulis akan menghasilkan proses *output* dari perintah yang sudah dijalankan pada *command line* (CMD) atau terminal, berikut ini adalah hasil *output* dari menjalankan fungsi koneksi dari

framework Aurora.JS ke database. Hasil dari *output* konfigurasi koneksi akan menghasilkan pesan seperti pada gambar berikut ini:

a. Pesan konfigurasi koneksi berhasil tanpa custom koneksi

```
C:\xampp\htdocs\aurora\master> node app.js
Aurora Serve on port 3000
```

b. Pesan konfigurasi berhasil koneksi dari *file custom* koneksi

```
C:\xampp\htdocs\aurora\master> node app.js
dhoni.config.js
Aurora Serve on port 3000
```

Hasil dari menjalankan perintah pada fitur *backup table* pada database, dengan cara menjalankan perintah db:*backup* untuk keseluruhan *table* pada database, db *backup* –t nama tabel database dapat dilakukan dengan *single table* atau *multi table backup* sesuai yang diinginkan pengembang, setelah menjalankan akan menghasilkan keluaran *output* berhasil menjalankan *backup table* pada database berikut ini dapat kita lihat di bawah ini :

a. Pesan table pada database berhasil di *backup*

```
C:\xampp\htdocs\aurora\master> aurora db:backup
Backup Table auroracoba1 Successfully On File
auroracoba1.json
Backup Table auroracoba2 Successfully On File
auroracoba2.json
Backup Table datakaryawan Successfully On File
karyawan.json
Backup Table data siswa Successfully On File
datasiswa.json
```

b. Pesan back*up single table* pada database berhasil di *backup*

```
C:\xampp\htdocs\aurora\master> aurora db:backup –
t datakaryawan
Backup Table datakaryawan Successfully On File
```

```
datakaryawan.json
```

c. Pesan *backup single table* pada database dengan *custom* koneksi berhasil di *backup*.

```
C:\xampp\htdocs\aurora\master>aurora db:backup –
t datakaryawan nama_custom_koneksi.config.js
Backup Table datakarywan Successfully On File
datakaryawan.json
```

d. Pesan *backup multi table* pada database berhasil di *backup*

```
C:\xampp\htdocs\aurora\master> aurora db:backup –
t datakaryawan,datasiswa
Backup Table datakarywan Successfully On File
datakaryawan.json
Backup Table datasiswa Successfully On File
datasiswa.json
```

e. Pesan *backup multi table* pada database dengan *custom* koneksi berhasil di *backup*.

```
C:\xampp\htdocs\aurora\master>aurora db:backup –
t datakaryawan,datasiswa nama_custom_koneksi.
Config.js
Backup Table datakarywan Successfully On File
datakaryawan.json
Backup Table datasiswa Successfully On File
datasiswa.json
```

Selanjutnya menjalankan perintah pada fitur *restore table* pada database,dapat dilakukan oleh pengembang dengan menjalankan perintah db:*restore* untuk *table* pada database, db *restore* -t nama *table* database dapat dipilih pengembang untuk melakukan *restore single table* atau *multi table*, hal ini terdapat syarat untuk

menjalankan *restore* database yaitu sudah melakukan perintah *backup* terlebih dahulu di *framework* Aurora.JS oleh pengembang. Berikut ini dapat kita lihat penjelasan fitur *restore table* pada database di bawah ini :

a. Pesan table pada database berhasil di *restore*

```
C:\xampp\htdocs\aurora\master>aurora
db:restore
Table datakaryawan successfully restore
Table datasiswa successfully restore
Table auroracoba1 successfully restore
Table auroracoba2 successfully restore
```

b. Pesan *restore single table* pada database berhasil di *restore*

```
C:\xampp\htdocs\aurora\master>aurora
db:restore –t datakaryawan
Table datakaryawan successfully restore
```

c. Pesan *custom* koneksi *restore single table* pada database berhasil di *restore*

```
C:\xampp\htdocs\aurora\master>aurora
db:restore –t datakaryawan nama_file_koneksi.
Config.js
Table datakaryawan successfully restore
```

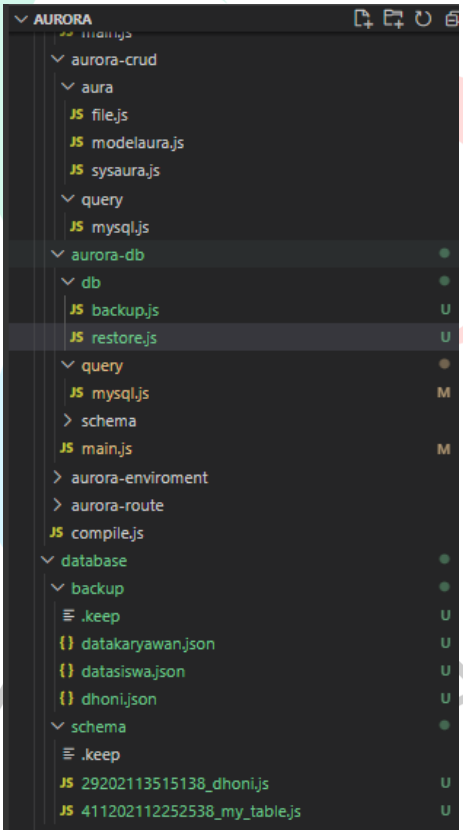
d. Pesan *restore multi table* pada database berhasil di *restore*

```
C:\xampp\htdocs\aurora\master>aurora
db:restore –t datakaryawan,datasiswa
Table datakaryawan successfully restore
Table datasiswa successfully restore
```

e. Pesan *custom* koneksi *restore multi table* pada database berhasil di *restore*

```
C:\xampp\htdocs\aurora\master>aurora
db:restore –t datakaryawan,datasiswa nama_file
_koneksi. Config.js
Table datakaryawan successfully restore
Table datasiswa successfully restore
```

Dari hasil *output* yang didapat ketika pengguna atau pengembang sudah melakukan apload data ke dalam github, tim pengembang aplikasi dapat memanfaatkan data tersebut ketika sudah melakukan *update* project sistem yang dirancang dan menjalankan perintah yang diinginkan untuk melakukan *backup* maupun *restore table*, dapat dilihat pada **Gambar 4.39.**



Gambar 4.39. Struktur *File* Modul *Backup* Dan *Restore Table* Pada *Framework* Aurora.JS

4.4 Perancangan Implementasi

Setelah melaksanakan tahap analisis dan perancangan yang telah dijalankan pada *framework* Aurora.JS, selanjutnya adalah tahapan implementasi yaitu tahapan dalam menjalankan sistem agar dapat

dijalankan atau dioperasikan sesuai dengan apa yang dirancang, tahap ini bertujuan untuk memastikan tercapai dan terlaksananya suatu tujuan. Pada instalasi Aurora.JS versi alpha membutuhkan software yaitu Node.JS dengan minimum versi 0.10 dengan memiliki *base* koneksi database yaitu MySQL. Dalam menjalankan *framework* Aurora.JS dibutuhkan sistem operasi yang dapat mendukung seperti windows, linux dan Mac OS. Penggunaan *Framework* Aurora.JS untuk menjalankannya dibutuhkan c*ommand line interface* (CMD) atau terminal. *Package framework* Aurora.JS juga sudah tersedia seperti perintah *node* dan NPM dalam menjalankan fitur yang tersedia pada *framework* Aurora.JS diperlukan installasi *package framework* Aurora.JS. Untuk saat ini pengembang masih melakukan secara *local* pada *framework* Aurora.JS

4.4.1 Testing

Setelah melaksanakan implementasi dilakukan kegiatan testing sistem yang diperiksa oleh penulis, testing bertujuan untuk melakukan cek sistem dalam memeriksa apakah terdapat *error* atau *bugs* yang yang dapat menggangu jalannya sistem. Pengujian dilakukan dengan memeriksa sistem pada *framework* Aurora.JS apakah berjalan dengan sesuai dari hasil *input* dan *output* yang dihasilkan. Setelah menjalankan proses testing pada *framework* Aurora.JS dan tidak ditemukan *bugs* atau *error*.

Pada tahapan ini penulis melakukan rancangan testing yang sudah dilakukan pada *framework* Aurora.JS. Dalam melakukan rancangan testing terdapat 2 metode yaitu *whitebox* dan *blackbox.* Metode pada *whitebox* testing merupakan pengujian yang didasarkan pada desain rinci perancangan menggunakan struktur control dari struktur ini digunakan untuk membagi pengujian ke dalam kasus-kasus yang terjadi pada program yang sedang dijalankan dan untuk menentukan seperti apa tampilan perangkat lunak berdasarkan *input* dan *output* suatu kode program. Sedangkan *black box* testing adalah pengujian yang dilakukan dengan cara

menganalisa hasil dari jalannya eksekusi program dan memeriksa fungsional setelah menggunakan program. *Black box* testing mengarah pada suatu tampilan sistem luar atau disebut *interface* pada suatu aplikasi, dengan tujuan agar lebih mudah digunakan oleh pengguna (Novitasari, 2020).

Dari hal ini untuk membantu penulis melakukan testing sistem dengan menggunakan metode *black box* untuk menguji fitur *backup* dan *restore table* pada database. Berikut ini adalah rancangan testing yang dilakukan oleh penulis dapat dilihat pada tabel **4.17.**

Tabel 4.17. Rancangan Testing

No	***Test Name***	***Test Steps***	***Expected Result***
1.	Membuat Koneksi Database	1. Mengakses *framework* Aurora.JS 2. Mengisi konfigurasi schema koneksi ke database 3. Menjalankan konfigurasi koneksi ke database	Konfigurasi koneksi sukses
2.	Menjalankan perintah pada terminal *backup table* pada database	1. Sudah terkoneksi ke database 2. Memiliki tabel pada database 3. Menjalankan perintah `db:backup` (untuk keseluruhan table pada database yang akan di backup) 4. Menjalankan perintah `db:backup nama_tabel -s` (nama schema/nama koneksi) 5. *File backup* tersedia di folder backup pada *framework* Aurora.JS	Menjalankan perintah pada *command line backup* sukses
3.	Menjalankan perintah pada terminal *backup single table* pada database	1. Menjalankan perintah pada terminal `db:backup nama_tabel` 2. Untuk menjalankan dengan custom koneksi dengan menambahkan `db:backup nama_tabel -s` (nama schema/nama koneksi) 3. *File backup single table* akan tersedia di folder backup pada framework aurora.js	Menjalankan perintah pada *command line backup single table* sukses
4.	Menjalankan perintah pada terminal *backup multitable* pada database	1. Menjalankan perintah *backup multi table* pada terminal `db:backup nama_tabel,nama_tabel -s` (nama schema/nama koneksi) 2. Jika berhasil *file backup multi table* akan tersedia di folder *backup* pada *framework* Aurora.JS	Menjalankan perintah pada *command line backup multi table* sukses

	Menjalankan schema *restore table* pada database	1. Sudah terkoneksi ke database 2. Sudah menjalankan perintah *backup table* dan *file* JSON tersedia pada *framework* Aurora.JS di folder *backup* 3. Folder *backup* akan mengembalikan data pada nama *file* yang diinginkan untuk di *restore*, dengan menjalankan `db:restore` untuk di *restore* ke dalam *table* pada database. 6. Untuk menjalankan *custom* koneksi menjalankan perintah schema `db:restore nama_tabel_database -s` (nama schema/nama koneksi)	Menjalankan perintah pada *command line restore* sukses
	Menjalankan perintah *restore single table* pada database	1. Menjalankan perintah pada terminal `db:restore nama_tabel` 2. Untuk menjalankan dengan *custom* koneksi dengan menambahkan `db:restore nama_tabel -s` (nama schema/nama koneksi) 3. *Read file* JSON pada folder *backup*, dan *write* data ke dalam *table*, *restore single table* akan tersedia di database.	Menjalankan perintah pada *command line restore single table* sukses
	Menjalankan *restore multi table* pada database	1. Menjalankan perintah pada terminal `db:restore nama_tabel,nama_tabel` 2. Untuk menjalankan dengan *custom* koneksi dengan menambahkan `db:restore nama_tabel,nama_tabel -s` (nama schema/nama koneksi) 3. *Read file* JSON pada folder *backup*, dan *write* data ke dalam *table*, *restore single table* akan tersedia di database.	Menjalankan perintah pada *command line restore multi table* sukses

UNIVERSITAS PEMBANGUNAN JAYA