# BAB II

# LANDASAN TEORI

## 2.1 Penelitian Terdahulu

Pada penelitian terdahulu, ditemukan sepuluh artikel jurnal yang membahas tentang *performance database*, baik yang bersifat *relational* maupun *non-relational*. Artikel-artikel tersebut membahas beragam aspek performa database dalam konteks berbeda-beda. Hasil dari kesepuluh artikel jurnal ini kemudian disusun dan dirangkum dalam Tabel 2.1 sebagai sumber referensi yang penting. Penyusunan tabel ini bertujuan untuk memberikan gambaran komprehensif tentang temuan dan perbandingan kinerja *database* dari perspektif berbagai penelitian sebelumnya. Tabel tersebut menjadi acuan utama dalam memahami kerangka penelitian dan konteks analisis *performance database* yang relevan.

Tabel 2. 1 Penelitian terdahulu

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| 1. | Analysis of Spatial Database Performance for Location Intelligence | Journal of Advanced Computing Technology and Application (JACTA)/ Vol. 2 No.2 | Safiza Suhana Kamal Baharin dan Patience Obiageli Akunne/ 2020 | *Response*/*Latency* *Throughput* berdasarkan CRUD | Mongo DB dan Oracle | 100.000 | *Response*/*Latency* *Throughput* | Hasil penelitian menunjukkan skalabilitas MongoDB lebih unggul dibandingkan Oracle, tapi Oracle lebih lebih unggul dari MongoDB tentang *agregat* sehingga *performance* Oracle lebih unggul dibandingkan MongoDB | Dengan membandingkan *response* dan *throughput* berdasarkan CRUD bisa digunakan untuk membandingkan *performance* DBMS |
| 2. | Performance analysis of NoSQL and relational databases with MongoDB and MySQL | Elsevier/ Vol. 24 No.3 | Benymol Jose dan Sajimon Abraham/ 2020 | *Command* query seperti simple select, insert, dan update | Mongo DB dan MySQL | 125.000 | *Response time for insert, update, and select* | Hasil penelitian yaitu ada perbedaan waktu eksekusi. *Database NoSQL* mendapatkan kinerja lebih | Dengan membandingkan *NoSQL* MongoDB dan MySQL berdasarkan *command* query bisa menghasilkan |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | baik dari *relational database* sehingga *NoSQL* lebih baik digunakan untuk perusahan yang ingin merubah databasenya dari konvensional. | *performance database*. |
| 3. | Analisis kinerja sinkronisasi database pada DBMS MySQL dan Oracle dengan menggunakan event-driven dan time-driven untuk pemantauan data cuaca | Jurnal of Aceh Physis Society (Sinta 3)/ Vol. 10 No.4 | Nurhani, Yuwaldi Away, dan Muhammad Syukri Surbakti/ 2021 | *Time Driven, Event Driven* | MySQL dan Oracle | 1000 | *Response time* | Hasil penelitian, menunjukkan bahwa dengan metode *Event-Driven* pada MySQL lebih mendapatkan data cuaca yang lebih akurat. Sementara Oracle dengan metode *Time driven* | DBMS Oracle lebih baik dibandingkan dengan MySQL dalam melakukan proses sinkronisasi untuk pemantauan data cuaca baik menggunakan metode *Time Driven* dan *Event Driven* |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | mampu mendapatkan data cuaca yang sesuai dengan data di MySQL. | |
| 4. | Performance Analysis of NoSQL and Relational Databases with CouchDB and MySQL for Application's Data Storage | Applied Sciences (Q2)/ Vol 10 Issue 23 | Cornelia A. Gy″orödi, Diana V. Dum¸se-Burescu, Doina R. Zmaranda Robert ¸ S. Gy″orödi Gianina A. Gabor dan George D. Pecherle/ 2020 | Menggunakan tiga aplikasi java yaitu IDEA, relasional MySQL, dan MySQL berbasis dokumen. Struktur digunakan ada 2 struktur yaitu CouchDB First Structure, dan Couch DB Second Structure. Penelitian juga menggunakan operasi CRUD | MySQL dan CouchDB | 1.000.000 | *Response time* | Hasil penelitian menunjukkan bahwa kinerja operasi CRUD. Pada penelitian ini ada menggunakan dua struktur data pendekatan CouchDB *non relational* yaitu pertama, dokumen berisi referensi lain dan kedua berisikan semua data dari setiap entitas. Struktur | Dengan membandingkan CouchDB dan MySQL yang mana CouchDB menggunakan 2 struktur data menghasilkan bahwa *performance* MySQL lebih baik dibandingkan CouchDB saat keadaan tertentu seperti *normalisasi database*. |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | CouchDB terbukti lebih efisien dibanding struktur kedua. Sementara Struktur kedua lebih baik dibandingkan MySQL *relational* saat melakukan operasi insert, select dan delete untuk jumlah data yang besar. MySQL *relational* menghasilkan kinerja lebih baik dibandingkan CouchDB keadaan tertentu seperti *normaliasi. database* dan | |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | prinsip ACID. | |
| 5. | Oracle 19c, SQL Server 2019, Postgresql 12 and MySQL 8 database systems comparison | Journal of Computer Sciences Institute/ Vol.17 | Arkadiusz Solarz, Tomasz Szymczyk/ 2020 | Metode eksperimental dengan mengukur waktu ekseskusinya yang menjalankan searching, grouping dan insert data | Oracle 19C, SQL Server, PostgreSQL, dan MySQL | 15.000.000 | *Response time* | Hasil penelitian menunjukkan bahwa ketika melakukan *data mining* maka Oracle menujukkan hasil yang cepat ketika memproses 500.000 record dibandingkan DBMS lainnya dan proses dari Oracle tetap masih stabil ketika record data mencapai 15 juta. Ketika melakukan data grouping maka Oracle dan SQL Server mencapai waktu kurang | Dengan melakukan perbandingan *performance database* Oracle 19c, SQL Server 2019, Postgresql 12 dan MySQL 8 maka mendapatkan kesimpulan bahwa Oracle menjadi DBMS yang tergolong baik setelah melakukan beberapa pengujian dan PostgreSQL berada di posisi kedua yang terakhir yaitu DBMS MySQL. |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | dari 500 ms ketika kurang dari 1 juta record data. Pengujian pada SQL server mendapatkan *performance* yang baik. Ketika melakukan *restore database* maka Oracle mengalami *performance* yang lama ketika mencapai 2 juta record data. Terakhir ketika melakukan searching data maka Oracle mendapatkat hasil *performance* yang baik hingga | |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | mencapai 15 juta record data DBMS kedua yaitu postgreSQL yang dapat memproses data 15 juta record data. | |
| 6. | Performance comparison of relational databases SQL Server, MySQL and PostgreSQL using a web application and the Laravel framework | Journal of Computer Sciences Institute/ Vol.17 | Rafał Wodyk, Maria Skublewska-Paszkowska/ 2020 | Query DML (Select, Insert, and Delete) | SQL Server, MySQL dan PostgreSQL | Up to 1000 records | *Response time* | Hasil penelitian menunjukkan bahwa ketika melakukan pengujian dengan 500 record data bahwa MySQL lebih unggul dibandingkan PostgreSQL. Sementara dengan 1000 record data SQL server dan MySQL menjadi DBMS yang lebih cepat dibandingkan PostgreSQL. | Dengan melakukan perbandingan DBMS SQL Server, MySQL, dan PostgreSQL menunjukkan bahwa MySQL mengalami kelambatan ketika jumlah record lebih dari 1000 record dibandingkan PostgreSQL. Ditambah lagi MySQL lebih cocok untuk aplikasi menengah dan rendah dibandingkan PostgreSQL. |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Sementara ketika melakukan pencobaan 5000 record data menunjukkan bahwa MySQL lebih lambat dibandingkan dengan DBMS lainnya. Terakhir melakukan percobaan dengan 20.000 record data PostgreSQL yang menujukkan lebih unggul dibandingkan DBMS SQL Server hanya beda 0.2 detik. | Oleh karena itu PostgreSQL menjadi DBMS yang baik untuk aplikasi menengah atau besar karena dapat menampung jumlah record data yang cukup besar ditambah lagi PostgreSQL memiliki keuntungan yaitu PostgreSQL *open source*. |
| 7. | Comparing Oracle and PostgreSQL, | Spinger Link/ Vol. 2 | Pedro Martins, Paulo Tom´e1, Cristina | Menggunakan metode eksperimental | Oracle, dan PostgreSQL | Up to 1.000.000 records | *Response time* | Hasil dari penelitian menggunaka | Dengan melakukan penelitian |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | Performance and Optimization | | Wanzeller, Filipe S´a1, and Maryam Abbasi/2021 | dengan *statement* Select COUNT(*) untuk mendapatkan informasi statistik dari tabel dan menggunakan perhintungan *standard deviation* dari hasil waktu ketika eksekusi | | | | n Oracle dan PostgreSQL ada beberapa tahapan yaitu tanpa optimasi yang dengan menjalankan 10 query mendapatkan nilai rata-rata eksekusi dari 2 DBMS yaitu Oracle memiliki kinerja yang lebik baik dibandingkan postgreSQL. Ketika melakukan pemrosesan dengan *primary key* dan *foreign key* maka ketika melakukan *standard deviation* maka postgreSQL | mendapatkan kesimpulan bahwa secara keseluruhan Oracle lebih stabil dan lebih cepat dalam mengeksekusi dari 10 query yang dipilih. Namun PostgreSQL lebih unggul ketika *optimization* dengan menghasilkan waktu peningkatkan rata-rata yang baik ketika melakukan pengujian. |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | lebih baik. terakhir ketika melakukan pemrosesan dengan index maka hasil dari rata-rata *standard deviation* Oracle tetap unggul dibandingkan postgreSQL. | |
| 8. | Comparison of MySQL, MSSQL, PostgreSQL, Oracle databases performance, including virtualization | Journal of Computer Sciences Institute/ Vol. 16 | Rafał Kleweka, Wojciech Truskowski, Maria Skublewska-Paszkowska/2020 | Metode eksperimental dengan menggunakan 4 skenario yaitu melibatkan eksekusi 1 query yang memilih data tanpa mempersempit kumpulan hasil. Skernario kedua yaitu melakukan pencarian dengan klausa Where dan skenario ketiga dengan memeriksa | Oracle, MSSQL, MySQL, dan PostgreSQL | Up to 100.000 records | Average Execution time | Hasil penelitan menunjukan bahas skenario pertama saat menggunakan perintah Select maka DBMS Oracle yang menjadi terbaik ketika eksekusi dan mendekati 1/ms dan MySQL dan MSSQL | Penelitian ini dapat disimpukan bawa setiap skenario DBMS memiliki keunggulan dan kelemahan tersendiri. Ketika skenario yang tidak memiliki kondisi WHERE ataupun yang memiliki klausa WHERE hasil data maka |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | efisiensi solusi database relasional yang dibandingkan dengan sub query yang berkorelasi. Skenario terakhir yaitu melakukan pengukuran dengan mengabungkan beberapa tabel dengan klausa Join dan Group By | | | | sangat lambat mencapai 700/ms dan PostgreSQL menghasilkan 298/ms. Pada skenario kedua dengan klausa WHERE menunjukkan MySQL menghasilkan hasil yang buruk dan Oracle menghasikan kecepatan yang sangat baik. Sementara MSSQL dan PostgreSQL memiliki hasil yang serupa. Skenario ketiga Oracle mendapatkan hasil yang sangat buruk dibandingkan | Oracle lebih unggul dibandingkan DBMS lainnya. Sementara dengan menggunakan subquery maka MSSQL yang lebih unggul. DBMS MySQL dan Oracle mengalami hasil yang buruk ketika menggunakan klausa Join dan Group By. Oleh karena itu, bisa disimpulkan bahwa Oracle memiliki kinerja yang terbaik dari sebagian skenario diuji dibandingkan DBMS lainnya. |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | DBMS lainnya. MSSQL berada posisi unggul untuk skenario ini dan MySQL berada di posisi ketiga. Pada skenario terakhir yang menggunakan klausa Join dan Group By ini Oracle dan MySQL menunjukkan hasil yang buruk dibandingkan MSSQL dan PostgreSQL. | |
| 9. | Performance analysis of relational databases MySQL, PostgreSQL and Oracle using Doctrine libraries | Journal of Computer Sciences Institute/ Vol. 24 | Marcin Choina, M. Skublewska-Paszkowska /2022 | Metode yang digunakan yaitu membandingkan waktu rata-rata dari hasil select, insert, update, dan delete | MySQL, PostgreSQL, dan Oracle | 100.000 | Average Execution time | Hasil dari penelitian menunjukkan bahwa dengan menggunakan kurang dari 100.000 record data maka | Dengan penelitian ini bisa disimpulkan bahwa Oracle berkerja lebih baik tanpa perlunya mapping ke database dan |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | PostgreSQL yang lebih unggul. Sementara Oracle lebih unggul dengan waktu yang singkat untuk memproses 100.000 record data dengan pengecualian *primary key*, dan grouping. MySQL melakukan proses record data dengan waktu yang lama dibandingkan DBMS lainnya. | PostgreSQL menjadi solusi tercepat sebuah aplikasi yang menggunakan *doctrine library*. |
| 10. | Performance analysis of selected database systems: MySQL, MS SQL, | Journal of Computer Sciences Institute/ Vol. 14 | Katarzyna Lachewicz/ 2020 | Metode yang digunakan yaitu membuat beberapa skenario pengujian untuk menguji dari *performance* | MySQL, MS SQL, dan PostgreSQL | 6000 | *Response time* | Hasil dari penelitian menunjukkan bahwa dari skenario yang telah dibuat dan | Dengan penelitian ini bisa simpulkan bahwa PostgreSQL adalah DBMS yang paling |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | PostgreSQL in the context of web applications | | | *database* untuk *single table* dan *related table* dengan menggunakan jumlah record mencapai 6000 record dan menggunakan aplikasi JMeter | | | | melakukan pengetesan dengan menggunakan beberapa *library*, *interface* dan ke DBMS secara langsung maka hasil yang didapatkan bahwa MySQL berkinerja lebih baik dibandingkan PostgreSQL dalam query yang lebih sederhana. Sementara untuk query yang lebih kompleks maka PostgreSQL yang lebih baik. MSSQL menujukkan | efisien untuk query yang kompleks untuk *web applications*. Sementara MySQL merupakan DBMS yang kinerja paling buruk. Kinerja DBMS sangat penting untuk kepuasan customer dalam mengakses *web application*. |

| No | Judul Artikel Jurnal | Nama Jurnal | Penulis / Tahun | Metode | DBMS | Jumlah Records | Parameter | Hasil | Kesimpulan |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | *database* yang efisien. | |

Dalam melakukan penelitian terdapat beberapa penelitian yang telah dilakukan sebelumnya dalam rentang waktu 2019-2022. Penelitian terdahulu menunjukkan hasil dari *performance* DBMS baik *relational database* ataupun *non-relational database*. Beberapa penelitian yang membandingkan performance dari Oracle dengan DBMS lainnya baik *relational* ataupun *non-relational.* Penelitian yang dilakukan menggunakan Oracle memperoleh hasil *performance* yang baik dengan jumlah record yang banyak meskipun ada beberapa pengujian seperti klausa JOIN membuat pemrosesan data di Oracle lebih lambat [3], [8], [10], [12], [13].

Sementara terdapat peneliti yang meneliti mengenai *non-relational database* yaitu MongoDB dan DBMS Oracle dan MySQL. Penelitian ini bahwa MongoDB tidak unggul daripada Oracle karena *throughput* yang dihasilkan lebih tinggi tapi MongoDB lebih unggul dibandingkan dengan MySQL [3], [7]. Penelitian meneliti mengenai DBMS MySQL dengan DBMS lainnya. Hasil penelitian MySQL terhadap beberapa *database* yang *non-relational* membuat MySQL lebih unggul tapi jika dibandingkan dengan *relational database*, MySQL memiliki *performance* yang lebih lambat dan tidak bisa lebih dari 20.000 record [7], [9]–[11], [13]–[15].

Penelitian meneliti mengenai PostgreSQL dengan DBMS lainnya. Hasil penelitian yang meneliti PostgreSQL bahwa PostgreSQL lebih baik dibandingkan DBMS lainnya kecuali Oracle yang memiliki *performance* yang masih lebih baik [10]–[15]. Penelitian meneliti mengenai MSSQL yang memiliki hasil penelitian yang masih baik dibandingkan MySQL tapi terdapat hasil bahwa memiliki *performance* yang hampir sama yang memiliki *performance* yang tidak jauh berbeda [10], [11], [13], [15].

Penelitian memiliki permasalahan yang diangkat dengan metode yang berbeda-beda seperti terdapat peneliti yang ingin mengetahui *response* database yang menggunakan query statement untuk mengenai location intellgence [3], untuk mengenai *performance database* baik secara *relational* ataupun *non-relational* ketika menyelidiki berbagai macam informasi yang

sangat besar dan terdapat penggunaan skenario [7], [9]. Selain itu terdapat penggunaan metode query statement yang menggunakan benchmark TPC-H untuk meneliti performance DBMS ketika memproses database yang semakin kompleks dan sangat banyak dengan menggunakan metode query statement [12].

Selain itu, terdapat penelitian yang meneliti DBMS dengan masalah ingin mencari mengetahui pemantauan data cuaca maka dari itu dibutuhkan proses penyimpan data melalui DBMS dan melakukan analisa dengan metode time-driven dan event-driven [8]. Peneliti yang meneliti Oracle dengan database lainnya untuk melakukan analisa DBMS yang berada 4 ranking teratas yaitu Oracle, MySQL, PostgreSQL dan MSSQL dengan menggunakan metode query statement [10], [13]. Terdapat peneliti yang meneliti DBMS karena ingin mengetahui performance database ketika diimplementasikan pada suatu aplikasi maka dari itu dibutuhkan penelitian performance DBMS ketika dijalankan di aplikasi dengan framework laravel dengan menggunakan bantuan query yang dijalankan di PHP [11]. Penelitian yang meneliti untuk menganalisis waktu response suatu database ketika menjalankan query statement dengan sebuah aplikasi [14], [15].

Dari penelitian terdahulu yang telah dijabarkan maka dalam penelitian ini ada beberapa mengadopsi penelitian terdahulu seperti penelitian yang menggunakan jumlah sample hingga 15.000.000 record [10]. Sementara penelitian ini hanya menggunakan 100.000 record data. Selain itu penelitian ini juga mengadopsi penelitian terdahulu yang menggunakan skenario query statement untuk menjadi sebuah skenario pengujian pada penelitian ini. Terakhir mengadopsi penelitian terdahulu yang menggunakan benchmark sebagai tolak ukur dari performance [12]. Pembeda penelitian ini dengan penelitian terdahulu yaitu menggunakan benchmark TPC-C sebagai tolak ukur performance database yang menganalisis data *Online Transaction Processing* (OLTP). Selain itu, menggunakan tool testing berupa JMeter yang diulang sebanyak 5 kali pengujian.

## 2.2 Tinjauan Teori

### 2.2.1 Pengertian Manfaat dan Type dari Database

#### 2.2.1.1. Pengertian Database

*Database* adalah suatu pengumpulan data atau informasi yang tersusun secara sistematis supaya bisa diolah sehingga nantinya bisa menghasilkan suatu informasi yang lebih akurat. *Database* juga adalah koneksi informasi yang telah tersusun rapi sehingga memudahkan untuk dikelola, *update*, dan diakses [16]. *Database* berfungsi untuk mempermudahkan identifikasi data. *Database* perlu diolah dengan suatu *software* untuk bisa mempermudah dalam melakukan pengelolahan data yang dikenal dengan *Database Management System* (DBMS). *Database Management System* (DBMS) memiliki fungsi utama yaitu untuk menyimpan dan mengamankan suatu data baik data perusahaan atau pribadi [17].

*Database* merupakan rangkaian data operasional dari suatu perusahaan yang telah terorganisir dan terintegrasi menggunakan metode-metode tertentu dalam komputer [18]. Hal ini memungkinkan penyediaan informasi optimal yang dibutuhkan oleh pengguna. *Database* memiliki beberapa manfaat yang bisa dirasakan oleh pengguna yaitu [19]:

1. Kecepatan dan kemudahan

   Manfaat pertama yang dirasakan yaitu kecepatan dan kemudahaan. Manfaat bisa dirasakan karena sistem *database* memberikan kemudahaan untuk menyeleksi data dari suatu kelompok secara berurutan dengan waktu yang singkat. Hal bisa melakukan pencarian suatu informasi dengan cepat. Kecepatan suatu *database* tergantung juga dengan *Database Management System* (DBMS) yang dipakai oleh pengguna.

2. Multi user

   Manfaat berikutnya yaitu *multi-user, Database* memberikan suatu kemudahan akses untuk pengguna saat menggunakan *database* tersebut secara bersamaan. Sistem juga memberikan suatu akses pada suatu dokumen ke *multi-user*. Hal ini

dikarenakan *database* disimpan pada satu server aja dan bisa diakses secara bersamaan.

3. Keamanan data

   Pada sistem *database* memiliki sistem keamanan data yang baik. Hal ini dikarenakan sistem database dengan bahasa pemrogramannya telah tersusun dengan *safety* melalui instrument *password* saat membuat data. Data yang telah dibuat bisa diberikan *role* yang bisa mengakses data tersebut. Oleh karena itu hanya beberapa pihak yang hanya bisa mengakses data. Keamanan data menjadi suatu hal yang diutamakan oleh sistem *database*.

4. Penghematan biaya perangkat

   Manfaat berikutnya yaitu penghematan biaya perangkat. Dengan adanya suatu sistem *database* yang terpusat maka setiap divisi di sebuah perusahaan memiliki *database* sehingga tidak memerlukan biaya untuk membeli perangkat seperti *hardware*.

5. Control data terpusat

   Control data terpusat adalah salah satu manfaat dari penggunaan *database* karena jika terjadi perubahaan pada suatu database maka hanya perlu mengubah satu *database* saja. Hal ini terjadi karena *database* berada disuatu *server* pusat sehingga pihak terkait bisa mengupdate data hanya satu *database* saja.

6. Meminimal Redundansi Data

   Dengan adanya sistem *database* maka tidak akan terjadi redundasi data. Redundasi data adalah penyimpanan suatu data yang sama didalam berkas yang berbeda. Oleh karena itu, jika salah satu data terupdate maka data lain tidak ikut terupdate. Oleh karena itu, dengan mengimplementasikan *database* bisa mengurangi redundasi data.

7. Integritas Data

   Sistem *database* tidak hanya mengurangi redundasi data tapi jika terdapat data yang sama maka data tersebut akan terintegrasi

dengan *database* yang telah ada. Integrasi data menjadi poin penting karena menjadi suatu keakuratan, konsistensi, aksesbilitas hingga kualitas suatu data. Jika integritas data semakin baik berarti kualitas, keakuratan data tersebut tinggi dan bisa digunakan untuk mendapatkan suatu informasi.

8. Independensi Data

   Independensi Data diartikan yaitu kemampuan untuk merubah suatu stuktur data tanpa membuat perubahan pada program yang sedang memproses data. Sistem *database* ini data-data tidak saling bergantung pada *software* karena *database* dibuat sesuai dengan kebutuhan informasi bukan berdasarkan *software*. Namun independensi juga tidak bisa diubah jika seseorang mengakses data tersebut.

9. Memudahkan untuk pembuatan aplikasi baru.

   Dengan adanya *database* membantu pengguna atau perusahaan untuk membuat sebuah aplikasi karena *database* telah dirancang dengan baik shinggaa perusahaan atau pengguna tidak perlu membuat *database* baru. Hal ini membantu *programmer* untuk membuat *user interface* aplikasi saja.

Dalam suatu *database* tentu memiliki beberapa komponen didalamnya. Komponen *database* memiliki fungsi masing-masing [19]. Komponen yang dimiliki *database* yaitu:

1. Data

   Data merupakan salah satu komponen di *database*. Data merupakan suatu kumpulan fakta-fakta dari objek yang ingin dimasukan ke dalam *database*. Data nantinya akan terintergrasi untuk menghilangkan redundansi data. Dengan data tersimpan di suatu *database* maka data tersebut bisa digunakan secara bersamaan oleh banyak pengguna (*User*). Terdapat 3 tipe data yang ada di *database* yaitu [20]:

   - Data operasional dari suatu organisasi

- Data masukan adalah data dari luar sistem yang dimasukan dengan keyboard dan bisa merubah data operasional
- Data keluaran adalah laporan melalui peralatan output sebagai hasil dari dalam sistem yang mengakses data operasional.

2. Hardware

   *Database* tentu membutuhkan *hardware* atau perangkat keras. *Hardware* yang digunakan *database* seperti *hard-disk* untuk dijadikan sebagai *second storage* dalam menyimpan data yang besar. Selain itu, *hardware* lainnya seperti *keyboard*, *mouse*, printer untuk mencetak data.

3. Software

   Database tidak hanya memerlukan *hardware* tapi juga membutuhkan *software* untuk sebagai penghubung antara *user* dan *database*. *Software* bisa diartikan sebagai *Database Management System* (DBMS) untuk membantu *user* dalam mengakses *database*.

4. User

   Dalam *user* database terdapat beberapa jenis *user*. Jenis-jenis *user database* yaitu:

   a. System Engineer

      *System Engineer* merupakan tenaga ahli yang mempunyai tanggung jawab untuk mengimplementasikan *database*. *System Engineer* juga memiliki tanggung jawab untuk melakukan peningkatkan dan melaporkan jika terjadi *error* pada sistem ke pihak yang bersangkutan.

   b. Database Administrator (DBA)

      *Database Administrator* merupakan pengguna yang memiliki tanggung jawab untuk bisa mengelola *database* dengan keseluruhan baik dari settingan, mencadangkan *database* saat ingin memindahkan *database* dari *database* satu ke database lain.

c. Programmer

Programmer memiliki tugas untuk membuat sebuah aplikasi dengan bahasa pemograman supaya *database* bisa diakses di aplikasi yang ingin dibuat.

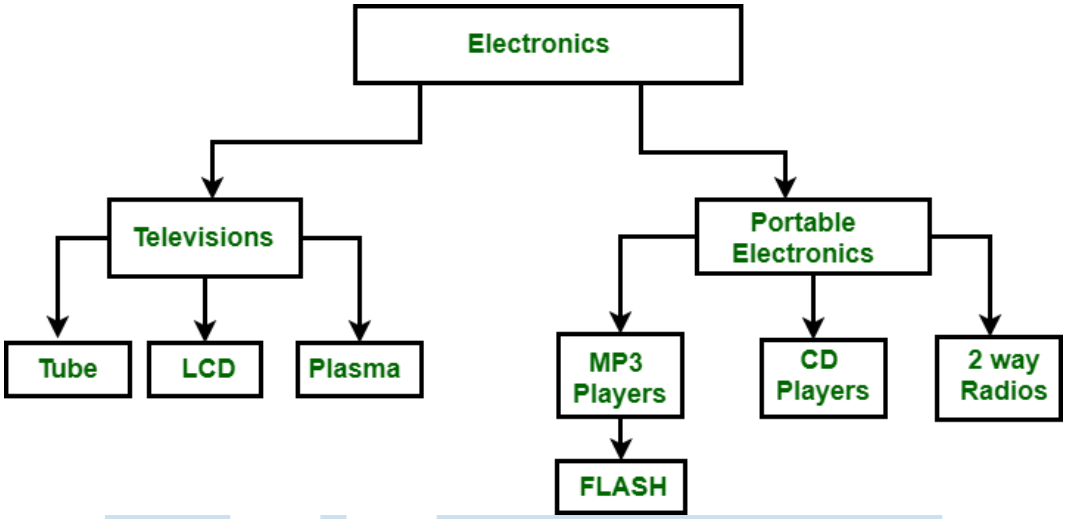
d. End-User

*End-User* bisa dikatakan sebagai pengguna yang dapat mengakses database dengan aplikasi yang telah dibuat sebelumnya oleh *programmer*. *End-user* juga bisa diartikan sebagai orang yang memakai *database* tanpa perlu mengelola sistem secara menyeluruh.

Selain *database* memiliki manfaat dan komponen yang terdapat di *database*. *Database* memiliki beberapa model yang dapat diterapkan. Model dalam *database* seperti *Hierachical Model*, *Network Model*, *Entity Relational Model*, dan *Relational Model*

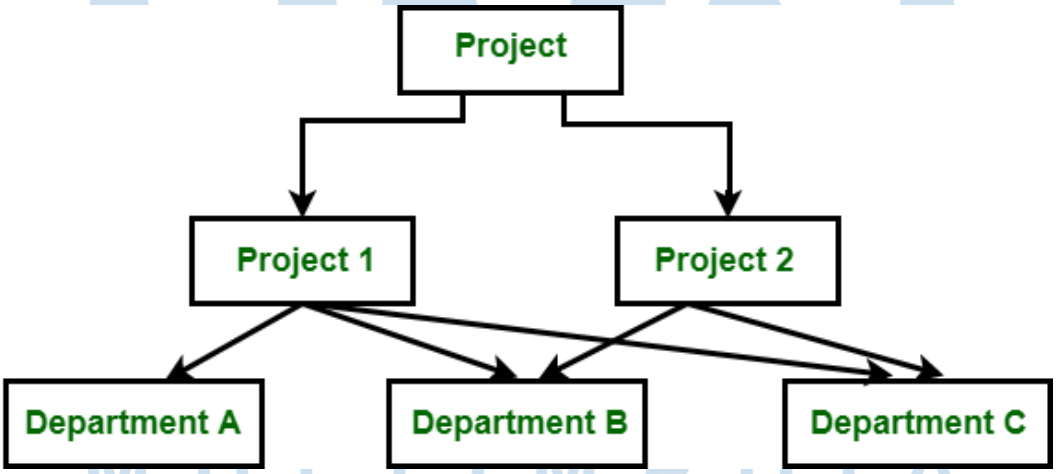
#### 2.2.1.2. Model Database

Model pertama dari *Database* yaitu *Hierarchical Model* (H-R Model). *Hierarchical Model* termasuk model *database* yang lama. *Hierarchical Model* dibuat oleh IBM pada tahun 1968. Data dari model *hierarchical* diatur dalam sebuah struktur pohon yang memiliki *root node* atau biasanya disebut dengan induk dan data lainnya berada di *child nodes*. Model ini memiliki *relational one-to-many* data. Hal ini bisa terlihat bahwa setiap *node* hanya memiliki 1 *child* hanya ada 1 *parent* dan 1 *parent* memiliki banyak *child* di dalam model ini [20], [21]. Contoh model dari *Hierarchical Model* terlihat seperti Gambar 2.1




Gambar 2. 1 Hierarchical Model Database
Sumber: [22]

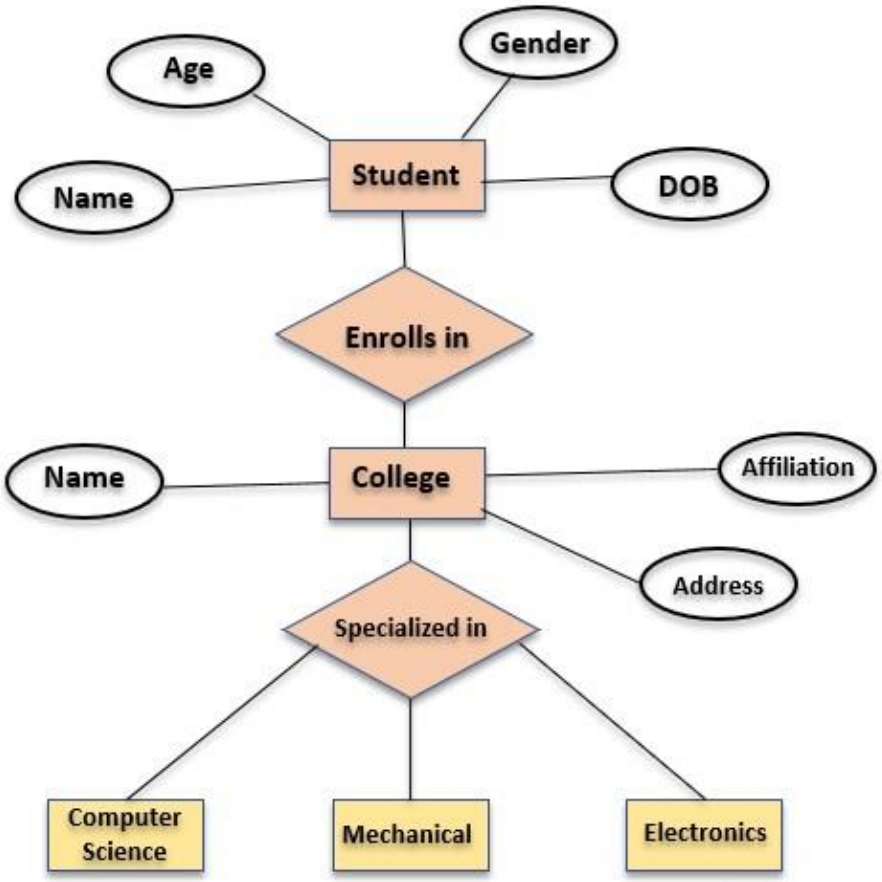
Model kedua yaitu *Network Model* merupakan perkembangan dari model sebelumnya yaitu *hirarchical model*. *Network Model* adalah model yang mengatur datanya seperti *graphic* dan model ini boleh lebih satu *parent node* sehingga data bisa diambil dari beberapa jalur. Relasi *Network Model* adalah *many-to-many* karena bisa mengakses banyak data ke data lainnya. Model *Network* menerapakan 2 konsep yaitu menggunakan *records* dan *set*. [20], [21]. Contoh model dari *Network Model* terlihat seperti Gambar 2.2




Gambar 2. 2 Network Model Database
Sumber: [22]

Model ketiga dari database yaitu *Entity Relationship Model*, Model *Entity Relationship Model* adalah model yang cocok untuk mendesain

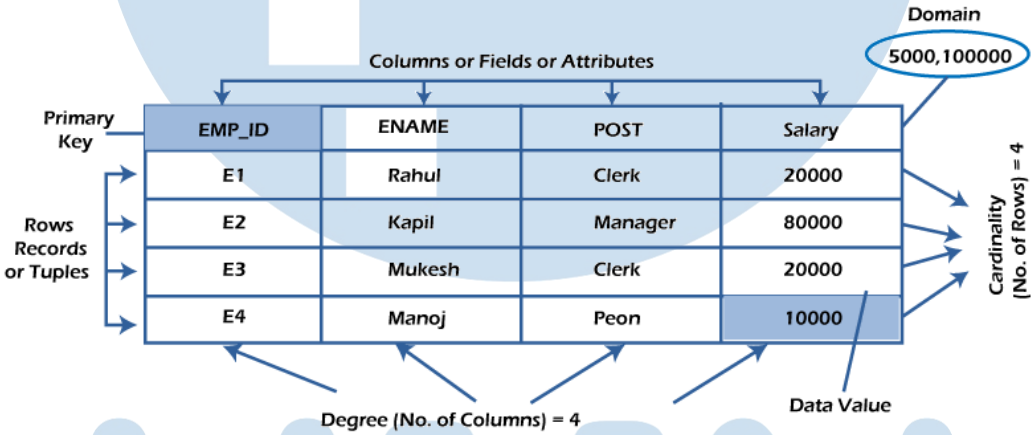
sebuah *database*. *Entity Relationship Model* terdapat entitas dan *atribute*. Entitas adalah object dalam sistem yang dimodelkan dan menyimpan informasinya. *Atribute* ada beberapa perbedaan seperti *simple attribute* berarti *atribute* ini tidak bisa dibagi lagi. Contoh dari *simple atrribute* yaitu no handphone. Ada juga *composite attribute* yang berarti *atribute* ini kombinasi dari lebih dari satu *atribute* sederhana. Selain itu, ada *derived attribute* adalah *atribute* turunan dari *atribute* lain. *Single-value Attribute* yang berarti berisikan nilai tunggal. Terakhir yaitu *Multi-value Attribute* yang berisikan lebih dari satu nilai. Contoh model dari *Entity Relational Model* terlihat seperti Gambar 2.3 [21]




Gambar 2. 3 Entity Relational Model Database
Sumber: [23]

Model terakhir yaitu *Relational Model* (R-Model). Model terakhir yaitu *Relational Database* adalah evolusi dari model *database* sebelumnya. *Relational Database* ini sekarang lebih sering digunakan karena model

*relational database* ini dikatakan lebih mudah dalam mengorganisasi data. Selain itu juga menyimpan dan melakukan pemrosesan data dengan baik. Konsep yang digunakan pada *database* ini yaitu menyimpan data dalam bentuk baris dan kolom. Baris yang ada di *Relational Database* disebut juga dengan record sedangkan kolom disebut dengan *field*. Terdapat aturan dan syarat yang mendasari pada *Relational Database* salah satunya yaitu Normalisasi yang berarti normalisasi menjadi satu aturan utama untuk merancang model *database* ini. Relasi yang digunakan pada *Relational Database* juga telah bertambah dari model sebelumnya seperti *one-to-one*, *one-to-many*, *many-to-many* [21], [24], [25]. Contoh model dari *Relational* model terlihat seperti Gambar 2.4




Gambar 2. 4 Relational Model Database
Sumber: [26]

*Relational Model* memiliki beberapa konsep yaitu:

1. *Table*: yang memiliki baris dan kolom. Kolom untuk mewakili *attribute* dan baris mewakili *record.*
2. *Tuple*: dapat diartikan sebagai baris dari sebuah *table.*
3. *Column*: dapat diartikan sebagai perwakilan dari himpunan nilai untuk atribut tertentu.
4. *Relation Schema*: berarti menggambarkan sebuah nama relasi dan atribut.
5. *Cardinality* berarti adalah jumlah tuple dalam sebuah relasi.
6. *Degree*: berarti jumlah atribut dalam suatu relasi atau *table*.

7. *Relation instance*.

### 2.2.1.3. Type dari penggunaan Database

Dari model-model *database* yang telah ada, mulai dari *Hierachical Model*, *Network Model*, *Entity Relational*, hingga terakhir *Relational Model*. Model-model ini membentuk beberapa tipe penggunaan *database*. Dalam *database* memiliki beberapa tipe penggunan *database* yang bisa digunakan. Tipe-tipe penggunaan *database* yaitu [3][18]:

1. Relational Database

   *Relational Database* adalah suatu tipe *database* yang sering digunakan saat ini, [3] *Relational Database* adalah *database* terstuktur yang berbentuk dalam tabel sehingga data bisa diatur ulang dan bisa diakses dengan beberapa cara yang berbeda-beda. *Relational Database* menggunakan bahasa query SQL. SQL ini dijadikan sebagai standar untuk interface baik pengguna atau program.

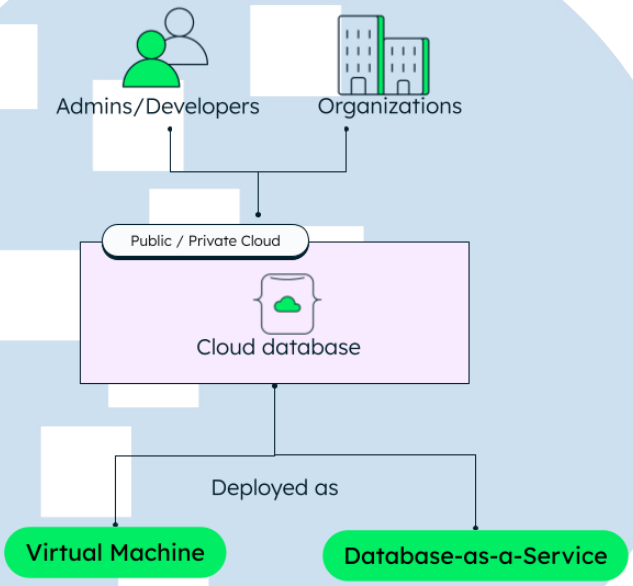
2. Distributed Database

   *Distributed Database* adalah *database* yang tersimpan pada lokasi tertentu. *Distributed Database* bisa dikatakan sebagai *database* untuk departemen atau kantor regional, kantor cabang, dan tempat kerja lainnya. *Database* bisa mencakup dua segmen yaitu *user database* dan operasional. Jenis *distributed database* bisa heterogen dan homogen.

3. Cloud Database

   *Cloud database* adalah *database* yang berbeda *relational database management system*. Hal ini bisa berbeda karena struktur pada *cloud database* ini lebih kompleks [27]. Ada beberapa *node* dari *database cloud* ini dibuat untuk layanan query, untuk pusat data yang letak datanya ada di lokasi geografis yang berbeda-beda sehingga ini bisa dikatakan bahwa *database* ini tersimpan di dalam

*cloud* dan bisa diambil saat terhubung internet dapat terlihat pada Gambar 2.5.




Gambar 2. 5 Cloud Database
Sumber: [28]

4. Object-oriented Database (Generasi 4)

*Object-Oriented Database* adalah *database* yang menyimpan data sebagai suatu objek. *Object-oriented Database* adalah kombinasi dari *Object-Oriented Programming* (OOP) dan prinsip *database*. *Object Oriented Database* adalah alternatif untuk menarik RDBS terutama di area aplikasi karena kinerja dari RDBS untuk menyimpan data sebagai objek dalam tabel tidak mencukupi. Fitur yang untuk dalam *Object Oriented Database* yaitu menyediakan dukungan untuk objek kompleks dengan menentukan struktur dan operasi yang diterapkan pada objek melakui *OOP interface*. *Object Oriented Database* membutuhkan interface program. Kemampuan beradaptasi terbesar dari *Object Oriented Database* ini menyesuaikan data untuk objek dan metode. Hal ini membuat *Object Oriented Database* dibuat dengan implementasi khusus dan tidak untuk tujuan umum seperti RDBS[29].

5. Graph Database (No-SQL)

*Graph Database* adalah jenis *database* yang menggunakan teori *graph* untuk menyimpan sebuah hubungan antara entitas dan relasi dalam suatu data dan query. Pada *graph database* setiap *node* menunjukkan sebuah entitas dan merepresentasikan dari setiap tepi hubungan antar *node*. Selain itu, *graph database* juga dimanfaatkan di berbagai aplikasi seperti aplikasi *social networking*, *security* and *access control*, *network and cloud management*. Dalam *graph database* ini ada beberapa fitur utama didalamnya yaitu[30]:

- Memudahkan untuk bisa mengidentifikasi hubungan antara data dengan menggunakan link/tautan.
- Hasil yang didapatkan saat menggunakan *graph database* yaitu *real-time*
- Kecepatan dalam *database* ini tergantung dengan jumlah hubungan antara elemen *database*.

6. NoSQL Database

*NoSQL Database* adalah *database* yang tidak memiliki struktur didalamnya. *Database* ini adalah sangat berguna untuk menyelesaikan suatu masalah kinerja data yang besar dan tidak bisa diselesaikan dengan *database* relasional [3]. *NoSQL Database* ini memiliki mekanisme untuk menyimpan dan mengambil data model dengan cara lain dibandingkan dengan hubungan tabel yang digunakan di dalam *database* relasional. *NoSQL* ini memiliki tujuan untuk membuat penyimpanan dan pemulihan data dengan mudah dan terlepas dari stuktur dan isinya [31]. Pada *database NoSQL* ada beberapa jenis [30]:

1) Document-based database

*Document based database* adalah *database* yang berbentuk *non-relational* dengan menyimpan data dalam baris dan kolom. *Document-based database* ini menyimpan data dalam dokumen XML, JSON, BSON. Dalam

*Document-based database* mempunyai elemen tertentu untuk bisa diakses dengan nilai index yang telah ditetapkan sehingga membuat pemanggilan kueri yang semakin lebih cepat. Dalam *Document-based Database* ada beberapa fitur seperti:

- *Flexible Schema*
- *Faster Creation and Maintenance*
- *No Foreign Keys*
- *Open Formats*

2) Key-Values stores

*Key-Values stores* adalah bentuk *database non-relational* yang sederhana karena setiap elemen data yang tersimpan di *database* akan dipasang *key-values pairs*. Hal ini membuat saat pemanggilan data maka akan menggunakan *unique key* yang telah dialokasikan untuk setiap elemen di *database*. Dalam *key-values stores* hanya ada dua kolom yaitu kunci dan nilai. Dalam *Key-Values stores* ada fitur utama yaitu:

- *Simplicity*
- *Scalability*
- *Speed*

3) Column-oriented databases

*Column-oriented database* adalah bentuk *database non-relational* yang menyimpan data dalam bentuk kolom saja. Hal ini membuat pengguna bisa membaca secara langsung tanpa perlu menghabiskan memori. *Coloumn-oriented database* dibuat untuk efisien dalam membaca data dan mengambil data dengan kecepatan yang tinggi. *Column oriented database* bisa menyimpan data dengan jumlah data yang besar. Fitur utama yang dimiliki oleh *Column oriented database* yaitu

- *Scalability.*
- *Compression.*
- *Very responsive*.

4) Graph-based database

7. Operational Database

*Operational Database* merupakan jenis *database* yang menyimpan data secara rinci yang diperlukan untuk mendukung operasi suatu organisasi atau perusahaan. *Operational database* bisa disebut juga dengan *subject-area database* (SADB). Contoh penerapan dalam *operational database* seperti *database* customer, *database* inventaris, *database* karyawan dan *database* yang mendukung operasi di suatu perusahaan.

8. Analytical Database

*Analytical Database* merupakan *database* yang menyimpan data dan informasi yang telah diambil dari operasional dan eksternal *database*. *Analytical Database* terdiri dari data dan informasi yang di ringkas dan paling dibutuhkan oleh sebuah organisasi manajemen dan *end-user*.

9. Data Warehouse

*Data Warehouse* adalah sebuah tempat penyimpanan data yang telah disimpan dari saat ini hingga beberapa tahun sebelumnya. Data yang disimpan berasal dari berbagai *database* operasional di dalam sebuah organisasi atau perusahaan. *Data warehouse* berfungsi sebagai tempat penyimpanan data yang telah diperiksa, diedit, dan terintegrasi, sehingga dapat digunakan oleh manajer dan pengguna akhir di seluruh organisasi profesional. *Data warehouse* terus berkembang hingga terakhir *data warehouse* digunakan sebagai *shared nothing architecture* untuk memfasilitasi *extreme scaling*.

10. Real-time Database

*Real-time Database* adalah sistem pengolahan data yang dirancang untuk menangani beban kerja yang dapat berubah secara cepat dan terus-menerus. Ini berbeda dari *database* tradisional yang berisi data yang tidak dipengaruhi oleh waktu. Contoh dari *Real-time database* adalah pasar saham, di mana harga saham berubah dengan cepat setiap saat. *Real-time database* ini bermanfaat dalam berbagai

bidang seperti akuntansi, hukum, catatan medis, sistem reservasi, dan analisis data ilmiah.

Suatu *database* dapat dijalankan dengan *Database Management System* (DBMS). *Database Management System* (DBMS) adalah komponen kunci dalam manajemen data modern dan memainkan peran penting dalam banyak aspek teknologi informasi dan komputasi. *Database Management System* (DBMS) membantu dalam menyimpan, mengelola, dan mengakses data dengan cara yang efisien dan terorganisir. Agar lebih memahami konsep *Database Management System* (DBMS) maka perlu meninjau beberapa aspek termasuk pengertian umum tentang *Database Management System* (DBMS), manfaat yang ditawarkan oleh sistem ini, komponen utama yang membentuk *Database Management System* (DBMS), dan contoh-contoh konkret dari berbagai jenis DBMS yang sering digunakan.

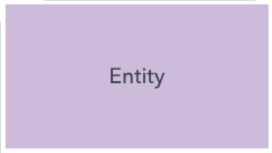
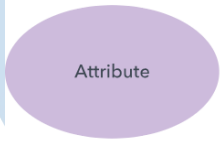
#### 2.2.1.4. Entity-Relationships Diagram

*Entity-Relationship Diagram* merupakan suatu diagram yang biasanya digunakan untuk mendesain suatu database [32]. Selain itu, *Enity-Relationship Diagram* merupakan diagram yang bentuk notasi grafis dalam pembuatan database menghubungkan antara data satu dengan data yang lain. *Enity-Relationship Diagram* memiliki 3 elemen dasar yaitu entitas, attribute, dan relasi. Selain itu, *Enity-Relationship Diagram* mempunyai beberapa relasi seperti *one-to-one*, *one-to many*, *many-to-many* [33], [34].

*Enity-Relationship Diagram* biasanya digambarkan pada 3 tingkat yaitu model data conceptual merupakan representasi tingkat tertinggi yang mencakup keseluruhan sistem dengan detail minimal, memberikan gambaran arsitektur sistem. Model data logis merupakan representasi lebih detail dibandingkan model data conceptual. Terakhir model data fisik merupakan perkembangan dari model data logis yang merepresentasikan ketika melakukan implementasi pada database [35]. *Enity-Relationship Diagram* terdapat *Conceptual Entity Relationship* dan *Physical Entity*

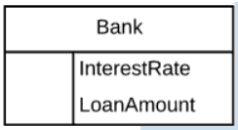
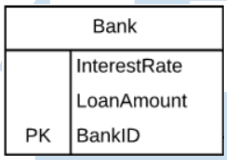
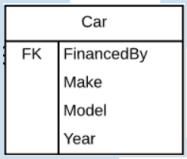
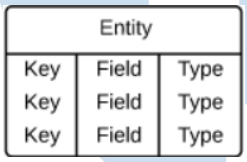
*Relationship*. *Conceptual Entity Relationship* digunakan untuk landasan *logical data model*. Selain itu, untuk membentuk hubungan kesamaan antara *Enity Relaltionship model* sebagai dasar intgrasi model data. *Conceptual Entity Relationship* memiliki simbol dapat terlihat pada Tabel 2.2 [36]:

Tabel 2. 2 Simbol Enity Relationship Diagram Conceptual

| Simbol | Keterangan | Description |
|---|---|---|
|  | Entitas | Entitas merupakan object yang mewakili suatu data penting dalam database. Entitas bisa berupa objek mengenai data yang dibutuhkan. |
|  | Atributte | Attribute merupakan karakteristik entitas yang membantu user untuk lebih memahami database. Sebuah Entitas perlu memiliki primary key sebagai ciri khas. |
|  | Relationships | Relationship digunakan untuk mendokumentasi dari interaksi 2 entitas |
|  | Link | Link digunakan untuk menghubungkan enitas dengan atributenya dan menghubungkan entitas dengan relationship |

Selain *Conceptual Entity Relationship*, Entity Relationship terdapat model *Physical Entity Relationship*. *Physical Entity Relationship* merupakan tingkat *Entity Relationship* yang sangat terperinci serta mewakilli proses penambahan informasi ke database. Pada *Physical Entity Relationship* akan menampilkan struktur tabel, termasuk nama kolom, tipe data, batasan, primary key, foreign key. *Physical Entity Relationship* memiliki simbol dalam pembuatan *Entity Relationship Diagram*. Simbol dapat terlihat pada Tabel 2.3 [36].

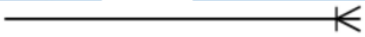
Tabel 2. 3 Simbol Entity Relationship Diagram Physical

| Simbol | Keterangan | Description |
|---|---|---|
|  | Field | Field mewakili sebuah tabel yang telah menentukan enitas dan atrribute dianggap sebagai kolom dalam database. |
|  | Primary Key | Primary Key merupakan attribute yang unik dalam satu entitas. Primary key untuk mengubungkan berbagai tabel dalam database satu sama lain secara efisien |
|  | Foreign Key | Foreign Key merupakan attribute yang memiliki relasi dengan entitas lain yang bisa berupa *one-to-one*, atau *one-to-many* |
|  | Types | Type merujuk pada tipe data pada attribute dalam entitas. Selain itu type juga dapat merujuk pada tipe entitas. |

Entity Relationship Diagram memiliki beberapa notasi seperti Bachman, IDEF1X, Barker's, Chen, Min-Max/ISO, Crow foot notation. Namun yang sering digunakan yaitu Crow foot notation. Notasi *Crow foot notation* dapat terlihat pada Tabel 2.4 [35], [36]

Tabel 2. 4 Notasi Crow's Foot Notation

| Simbol | Keterangan |
|---|---|
|  | One |
|  | Many |
|  | One (and only one) |

| Simbol | Keterangan |
|---|---|
|  | Zero or one |
|  | One or many |
|  | Zero or many |

Tabel 2.4 menggambarkan notasi simbol dari *Crow's Foot Notation* yang digunakan dalam Entity-Relationship Diagram (ERD). Simbol "One" digunakan untuk menandakan hubungan satu ke satu antara entitas, sementara "Many" mengindikasikan hubungan satu ke banyak. Simbol "One (and only one)" menunjukkan bahwa setiap entitas pada satu sisi hubungan harus terhubung dengan satu dan hanya satu entitas pada sisi lainnya. Simbol "Zero or one" menyatakan bahwa setiap entitas pada satu sisi hubungan dapat terhubung dengan nol atau satu entitas pada sisi lainnya. "One or many" menunjukkan bahwa setiap entitas pada satu sisi hubungan dapat terhubung dengan satu atau banyak entitas pada sisi lainnya. Terakhir, "Zero to many" mengindikasikan bahwa setiap entitas pada satu sisi dapat terhubung dengan nol atau banyak entitas pada sisi lainnya.

### 2.2.2 Pengertian Manfaat, dan Komponen *Database Management System* (DBMS)

#### 2.2.2.1 Pengertian Database Management System (DBMS)

*Database Management System* (DBMS) merupakan sebuah *software* atau sistem yang sudah mendapatkan rancangan secara khusus untuk melakukan pengelolaan pada suatu tempat penyimpanan data. Selain itu,

*Database Management System* (DBMS) sebagai perantara antara *user* dan *database*. Menurut Beal pada tahun 2017 di [20], Untuk bisa memakai *Database Management System* (DBMS) ini dapat menggunakan bahasa *database* yang telah ditentukan oleh perusahaan *Database Management System* (DBMS). Bahasa *Database Management System* (DBMS) terkadang berisikan intruksi-intruksi yang telah difomulasikan sehingga instruksi ini bisa diproses oleh *Database Management System* (DBMS). *Database Management System* (DBMS) ini memiliki tujuan untuk membantu dalam pengelolaan, pemeliharaan data dalam jumlah data yang besar. Selain itu, *Database Management System* (DBMS) juga bertujuan untuk tidak bisa menimbulkan kekacauan dan bisa dipakai oleh *user* sesuai kebutuhan *user*.

Ada beberapa pengertian *Database Management System* (DBMS) menurut para ahli yaitu, *Database Management System* (DBMS) adalah suatu *software* yang melakukan pemantauan terhadap semua akses *database* untuk melayani kebutuhan-kebutuhan penggunanya. Tidak hanya itu, *Database Management System* (DBMS) dapat diartikan sebagai suatu manajemen yang memiliki keefektifan dalam melakukan pengorganisasian sumber daya data. Selain itu, *Database Management System* (DBMS) adalah sebuah gabungan dari *database* yang menggunakan sistem manajemen berbasis data di komputer [20], [37]. Selain itu, terdapat ahli yang mengartikan *Database Management System* (DBMS) adalah suatu *software* yang mengizinkan *user* untuk mendefinisikan, menjaga, membuat dan memiliki akses *database* [38]. Sementara terdapat ahli yang mengartikan DBMS sebagai program yang menyediakan akses ke *database* yang akan mempermudah dalam segala proses penyimpaan dan pengamanan data tersebut [39].

#### 2.2.2.2 Manfaat Database Management System (DBMS)

Setelah memahami pengertian *Database Management System* (DBMS) dari beberapa ahli, jelas bahwa *Database Management System* (DBMS) tentu memiliki beberapa manfaat yang bisa dirasakan oleh *user* saat menggunakan *Database Management System* (DBMS). Manfaat dari *Database Management System* (DBMS), yaitu [20]:

1. Database dalam DBMS bisa digunakan secara bersamaan

   Manfaat yang pertama yang dirasakan dari DBMS yaitu jaringan *database* yang dipakai ini bisa digunakan secara bersamaan. Dengan ini akan menghemat tempat penyimpanan karena tidak membutuhkan banyak tempat saat digunakan secara bersama.
2. Proses akses DBMS dapat terlaksana dengan mudah dan cepat.

   Manfaat berikutnya yaitu proses dalam mengakses data bisa digunakan dengan mudah dan cepat sehingga tidak memerlukan waktu yang lama untuk bisa menyelesaikan dan bisa lebih cepat dan tepat. Dengan menggunakan *Database Management System* (DBMS) ini akan mengakses data-data penting dilakukan dengan mudah dan bisa mengaksesnya dalam waktu yang bersamaan
3. Menghemat ruang dan penyimpanan DBMS

   *Database Management System* (DBMS) memiliki keunggulan yaitu penyimpanan yang besar sehingga bisa menghemat ruang penyimpanan. Selain itu *Database Management System* (DBMS) juga bisa memperbaiki integritas data.
4. Menjaga keamanan data-data penting

   Manfaat berikutnya yaitu Sistem keamanan dalam DBMS ini bisa dikatakan baik karena bisa *recovery* dan *backup* untuk bisa meminimalkan kehilangan data. Oleh karena itu, semua file dan isi akan baik-baik saja walaupun digunakan dalam jangka waktu yang panjang.

5. Mencegah dan Menghilangkan hasil duplikasi maupun inkonsistensi data

   Manfaat lainnya adalah kemampuan *Database Management System* (DBMS) untuk secara otomatis menghapus hasil duplikasi dan mengatasi inkonsistensi data. Ini dimungkinkan oleh kemampuan *Database Management System* (DBMS) untuk mempertahankan independensi data dan melakukan perubahan struktur program.

6. DBMS menangani data dalam jumlah besar.

   Manfaat terakhir yaitu *Database Management System* (DBMS) memiliki kemampuan untuk integrasi dan mempertahankan file dalam keadaaan valid dan konsisten walaupun tersimpan pada jumlah data besar.

#### 2.2.2.3 Komponen-Komponen Database Management System (DBMS)

Dari mengetahui manfaat yang diberikan oleh *Database Management System* (DBMS) maka perlu mengetahui komponen-komponen yang terdapat dalam *Database Management System* (DBMS). Dalam *Database Management System* (DBMS) ada beberapa komponen-komponen baik fungsional atau modul. Komponen yang terdapat di DBMS yaitu [20]:

1. File Manager

   *File Manager* merupakan komponen yang bertugas melakukan pengelolaan ruang yang tersedia, khususnya di disk, serta struktur data untuk keperluan presentasi dan menyediakan informasi mengenai data yang tersimpan dalam memori.

2. Database Manager

   *Database Manager* merupakan komponen yang berfungsi sebagai *interface* antara tingkat data rendah yang tersimpan dalam database dan aplikasi serta query yang digunakan dalam sistem manajemen tertentu. Pembuatan komponen ini bertujuan untuk menyediakan tampilan data yang konsisten antar tingkat data

sesuai dengan basis datanya. Selain itu, *Database Manager* mampu menyajikan informasi yang berguna karena *Database Management System* (DBMS) secara khusus dirancang untuk pemeliharaan dan pengelolaan file dalam jumlah data yang besar.

3. Query Processor

   *Query Processor* merupakan komponen yang bertanggung jawab untuk menerjemahkan perintah yang diberikan oleh pengguna menjadi query yang dapat dipahami oleh *Database Management System* (DBMS). Fungsinya adalah mengubah perintah dalam bahasa manusia menjadi instruksi level rendah sehingga DBMS dapat memprosesnya dengan efisien. Dengan demikian, *Query Processor* dapat dianggap sebagai penerjemah perintah dari bahasa manusia ke bahasa mesin dengan tingkat level rendah, memfasilitasi komunikasi yang efektif antara pengguna dan *database*.

4. Data Manipulation Language (DML) Precompiler

   *Data Manipulation Language* (DML) adalah komponen yang mengkonversikan perintah dari *Data Manipulation Language* (DML). *Data Manipulation Language* (DML) bertujuan untuk membantu sistem manajemen *database* lebih mudah dalam memahami bahasa dari pengguna. *Data Manipulation Language* (DML) ada dampak positifnya yaitu hasil dari perintah yang diberikan akan lebih cepat terselesaikan dengan menggunakan sistem database tersebut.

5. Data Definition Language (DDL) Compiler

   *Data Definition Language* (DDL) merupakan komponen yang mengkonversikan mengenai berbagai macam perintah pada kumpulan tabel berisi dari informasi mengenai data baik data yang bersifat penting hingga biasa saja. Komponen ini bisa digunakan untuk membuat indeks atau mengubah format bentuk sebelumnya dan itu akan disimpan di kamus data.

6. People

   *People* diartikan sebagai *user* yang terlibat dalam sebuah sistem. Pengguna akan melakukan aktivitas di manajemen *database* mulai dari masukan data, mengelolah, menyimpan dan melakukan pengawasan.

7. Hardware

   *Hardware* merupakan komponen dari *Database Management System* (DBMS) dan tidak hanya DBMS tapi semua aplikasi juga membutuhkan hardware untuk menjalankan aplikasi.

8. Software

   *Software* merupakan elemen yang diperlukan oleh *Database Management System* (DBMS) agar dapat beroperasi. Komponen ini terdiri dari beberapa bagian, termasuk sistem operasi, perangkat lunak DBMS itu sendiri, dan program aplikasi yang memanfaatkannya.

9. Data

   Data merupakan komponen yang menjadi penghubung antara komponen *software* dan *user*. Semua yang dimasukkan dalam *Database Management System* (DBMS) adalah data, baik itu data berupa dokumen, gambar, animasi, atau bentuk data lainnya.

10. Prosedur.

    Komponen terakhir yaitu prosedur yang berarti aturan-aturan yang digunakan untuk menjalankan *Database Management System* (DBMS). Hal ini membuat pengguna sistem akan melakukan pengelolaan database dengan prosedur tertentu untuk mendokumentasikan atau menjalankan sistem tersebut.

#### 2.2.2.4 Database Management System (DBMS) Languages

Dalam menjalankan database pada *Database Management System* (DBMS) maka dibutuhkan beberapa *languages* yang digunakan. *Database Management System* (DBMS) tidak ada pemisahan level yang ketat hanya satu bahasa yang disebut dengan *Data Definition Language* (DDL). *Data Definition Language* (DDL) biasanya digunakan oleh *Database Administrator* (DBA) dan peracang *database* untuk bisa mendefinisikan kedua *schema*. *Database Management System* (DBMS) ini memiliki *compiler Data Definition Language* (DDL) yang berfungsi untuk memproses *Data Definition Language* (DDL). Selain itu ada juga bahasa lain yaitu *Storage Definition Language* (SDL), bahasa ini digunakan untuk menentukan *schema* internal. Pada sekarang ini kebanyakan DBMS *relational* tidak ada bahasa khusus yang menjalankan peran *Storage Definition Language* (SDL). Untuk arsitektur tiga *schema* memerlukan *View Definition Language* (VDL), untuk menentukan tampilan pengguna dan pemetaannya ke *schema* konseptual tetapi di ada beberapa *Database Management System* (DBMS) yang menggunakan *Data Definition Language* (DDL) untuk mendefinisikan *schema* konseptual dan eksternal. *View Definition Language* (VDL) digunakan untuk mendefinisikan tampilan pengguna atau aplikasi sebagai hasil *query*[40].

Selain itu, *Database Management System* (DBMS) menyediakan satu set operasional atau bahasa yang disebut juga dengan *Data Manipulation Language* (DML) yang memiliki tujuan untuk memanipulasi database. Manipulasi seperti pengembalian, penyisipan, penghapusan, dan modifikasi data pada *Database Management System* (DBMS). *Data Manipulation Language* (DML) ada dua jenis yaitu *Data Manipulation Language* (DML) *high-level* atau *nonprocedural* DML dan *Low-level* atau *procedural*. DML *high-level* berfungsi untuk menentukan operasi *database* yang kompleks secara ringkas. *Data Manipulation Language* (DML) *high-level* untuk dimasukkan secara interaktif dari terminal yang disematkan dalam bahasa pemograman. *Data Manipulation Language*

(DML) *low-level* atau *procedural* harus tertanam dalam bahasa pemrograman tujuan umum. *Data Manipulation Language* (DML) *low-level* juga disebut *record-at-a-time* karena termasuk dalam *property*. *Data Manipulation Language* (DML) *high-level* yang digunakan secara interaktif mandiri disebut juga bahasa query.

#### 2.2.2.5 Contoh-contoh DBMS

Dalam DBMS ada beberapa aplikasi untuk melakukan pengelolaan baik data besar seperti database perusahaan, *database* universitas dan lainnya. Ada beberapa *Database Management System* (DBMS) yang sering digunakan seperti [20] :

1. DBMS MySQL

   *Database Management System* (DBMS) yang pertama yaitu MySQL, DBMS ini memiliki banyak pengguna karena *Database Management System* (DBMS) MySQL DBMS yang gratis. *Database* ini bersifat *open-source* menggunakan model *client server*. Keamanan data dari *database* MySQL cukup baik dan kecepatan pengaksesan data juga selalu stabil. *Database Management System* (DBMS) ini ada kekurangan yaitu masih kurang kompatibel dengan bahasa Foxpro, Visual Basic, dan Delphi.

2. Oracle

   *Database Management System* (DBMS) berikutnya yaitu Oracle, *Database Management System* (DBMS) Oracle merupakan DBMS berbayar dan bagus, Pada Oracle ini memiliki banyak fitur yang bisa memberikan pemenuhan tuntutan fleksibilitas terhadap perusahaan besar. *Database Management System* (DBMS) Oracle juga memiliki proses transaksi dengan performa yang tinggi. Dengan kemampuan yang tinggi, hal ini yang membuat *software* menjadi sangat mahal ditambah dengan komputerisasi yang cukup rumit sehingga pengguna tidak perlu

meragukan sisi keamanan dari *Database Management System* (DBMS) Oracle.

3. Microsoft SQL Server

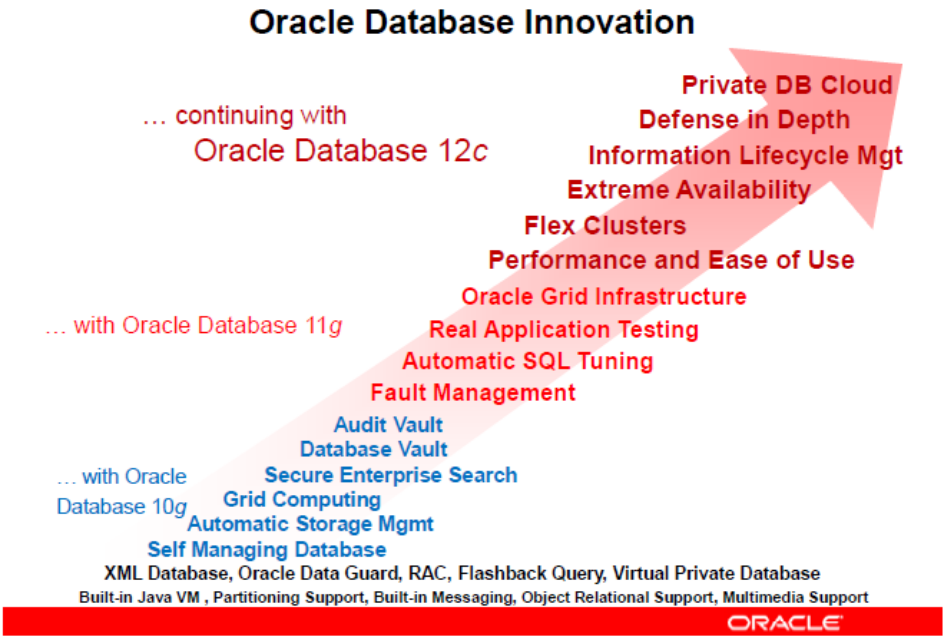
   Microsoft SQL Server ini *Database Management System* (DBMS) yang cocok untuk mengaplikasikan sistem pada jaringan komputer terutama dalam perusahaan-perusahaan besar karena memiliki kemampuan untuk bisa melakukan pengelolaan terhadap data dengan jumlah yang besar.

### 2.2.3 ORACLE


Gambar 2. 6 Logo Oracle
Sumber: [41]

#### 2.2.3.1 Perkembangan Oracle

Oracle adalah *database* yang sudah berkembang dari tahun 1977 oleh perusahaan Oracle Corporation dan sekarang ini Oracle telah menjadi *Database Management System* (DBMS) yang no 1 dibanding DBMS lainnya. Oracle telah menambahkan lebih banyak kekuatan dan fitur data serta pengelolaannya [42], [43]. Oracle *Database Inmovation* bisa dilihat dari Gambar 2.7 Oracle *Database Innovation*.



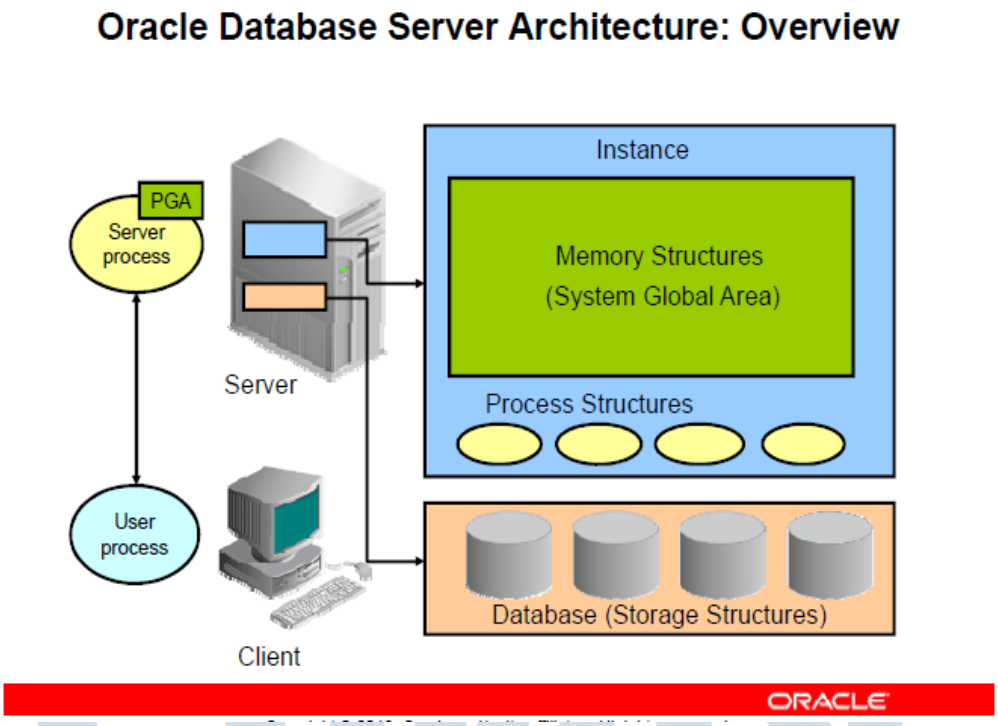

Gambar 2. 7 Oracle Database Innovation
Sumber: [42]

Sekarang Oracle telah mencapai 12c. pada Oracle *database* 10g adalah sistem manajemen *database* pertama yang dirancang untuk komputasi grid. Perkembangan berikutnya Oracle *database* 11 g yang mengkonsolidasi dan memperluas kemampuan unik Oracle untuk memberikan manfaat komputasi grid, mengubah pusat data dari silo sumber daya sistem yang terisolasi menjadi kumpulan server dan penyimpanan yang sama. Oracle *Database* 12c dan *Enterprise Manager Cloud* Control dirancang untuk *cloud computing*. *Cloud Computing* memberikan manfaat bagi suatu perusahaan karena data perusahaan bisa tersimpan di dalam *cloud*. Manfaat utama dari penggunaan *cloud computing* [42], yaitu:

- Mengurangi penyebaran *server* dan tingkatkan penggunaan CPU dengan mengkonsolidasi *server* yang lebih sedikit.
- Meminimalkan waktu bagi *Database Administrator* (DBA) untuk menginstal dan mengkonfigurasi *database* dengan mengotomatisasi penyebaran konfigurasi *database* standar
- Memperkirakan kebutuhan sumber daya dimasa depan dengan menganalisis laporan

Kebanyakan perusahaan menggunakan Oracle untuk menyimpan data, pemrosesan data, pemrosesan transaksi dan analisis binsis. Oracle memiliki banyak alat untuk mengelola database seperti alat untuk akses database *built-in* yaitu SQL * Plus, Oracle Enterprise Manager Cloud Control biasanya digunakan untuk administrasi, SQL Developer ini mirip dengan Microsoft SQL Server Management Studio yang dapat digunakan untuk mengembangkan dan manajemen *database*.

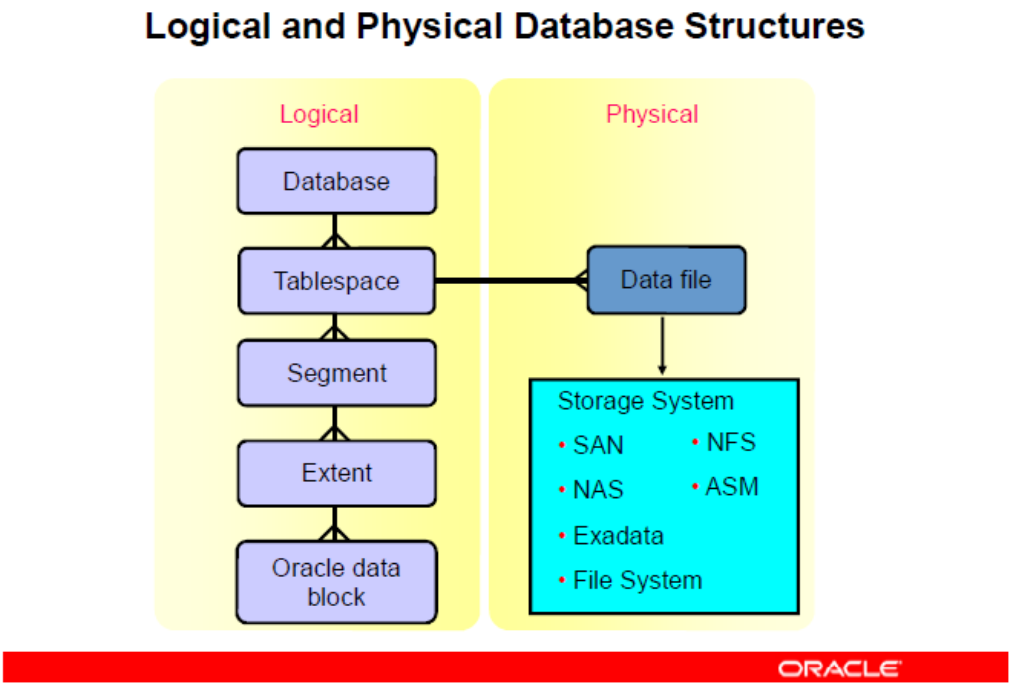
### 2.2.3.2 Struktur & Architecture DBMS Oracle




Gambar 2. 8 Oracle Database Server Architecture: Overview
Sumber: [42]

Gambar 2.8 adalah *architecture* dari Oracle. Dalam *architecture server* Oracle ada 3 struktur utama yaitu ada struktur memori, struktur proses dan struktur penyimpanan. Pada dasarnya sistem *database* Oracle terdiri dari *database* Oracle dan instance *database*. *Database* ini ada dibagi menjadi struktur logis dan struktur fisik. Struktur fisik dan logis ini struktur yang terpisah. Penyimpanan fisik data bisa dikelola tanpa mempengaruhi akses

ke struktur penyimpanan logis. *Instance* terdiri dari struktur memori. Memori bersama disebut juga dengan *System Global Area* (SGA) [42].




Gambar 2. 9 Logical dan Physical Database Structures
Sumber: [42]

Gambar 2.9 merupakan *logical* dan *physical database structure* dari Oracle. DBMS Oracle memiliki *logical structure* dan *physical structures*. *Database, Tablespace* dan *File* data ini memiliki hubungan dengan *database* karena dalam *database* Oracle dibagi menjadi 2 atau lebih *tablespace*. 1 atau lebih file data eksplisit ini dibuat untuk setiap *tablespace* untuk secara fisik menyimpan data dari semua segmen dan *tablespace* [42]. Jika *tablespace* bersifat sementara maka file yang didalam *tablespace* memiliki file sementara bukan file data. Data Block adalah data *database* Oracle yang disimpan dalam blok data dan satu blok ini sesuai dengan jumlah byte tertentu dari *physical space* pada disk. Ukuran blok data bisa diatur saat membuat sebuah *tablespace*.

### 2.2.4 My SQL

Gambar 2. 10 Logo MySQL
Sumber: [44]

#### 2.2.4.1 Perngertian My SQL

MySQL adalah perwakilan *dari Database Management System* yang dibangun menggunakan mesin yang berbeda. Kelebihan dari MySQL selain mudah digunakan, ada kelebihan lain yaitu kinerja yang stabil karena MySQL bisa menangani data yang ukurannya cukup besar. MySQL juga aman. MySQL diciptakan oleh 2 orang dari Swedia dan 1 Finlandia. Selain ada kelebihan MySQL juga ada kelemahan yaitu setelah MySQL dibeli dengan Oracle pada tahun 2009 membuat data nya tidak *open-source* dan MySQL sekarang ini ada yang gratis tapi fitur yang diberikan terbatas dan berbayar. MySQL ini telah memiliki 100 juta pengguna di seluruh dunia. MySQL banyak digunakan dari perusahaan yang menengah kecil hingga besar. Contoh dari perusahaan yang memakai MySQL pada sistem *database* yaitu Yahoo!, Youtube, WordPress.[4], [45].

#### 2.2.4.2 Struktur DBMS My SQL

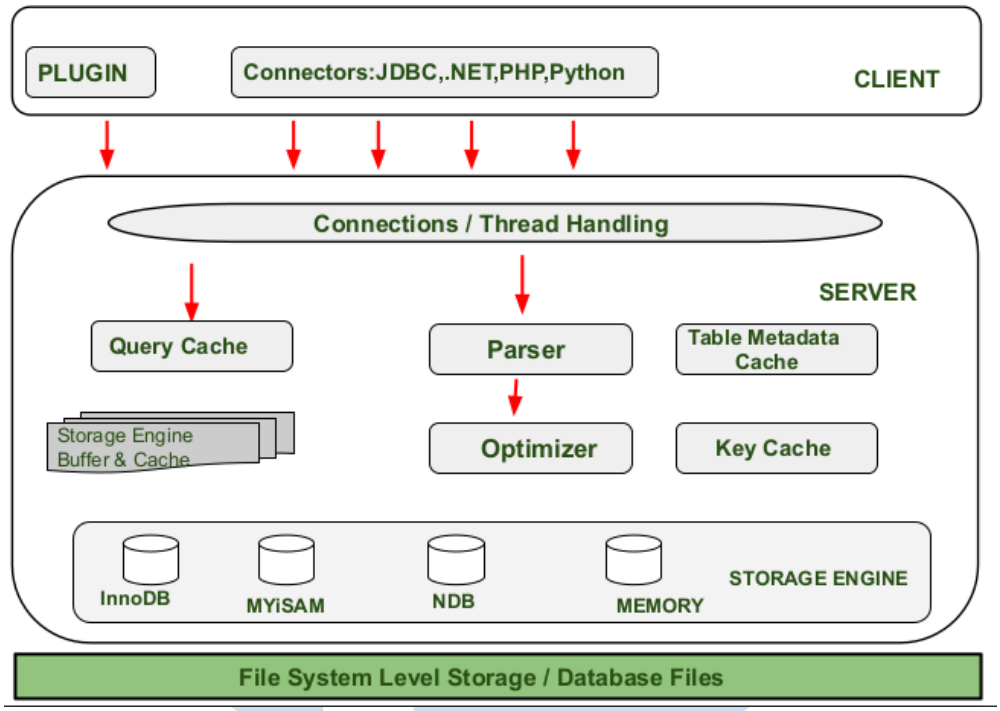
MySQL menggunakan *Data Definition Language* (DDL) yang termasuk dalam perintah query di SQL. *Data Definition Languange* (DDL) membuat *user* bisa membuat table, mengubah dataset, menghapus data, membuat indeks dan menentukan struktur penyimpanan tabel. Dalam *Data Definition Language* (DDL) ini ada 5 perintah yaitu CREATE, RENAME, ALTER, SHOW dan DROP.

CREATE memiliki perintah untuk membuat *database* baru mulai dari table ataupun kolom. ALTER adalah perintah untuk melakuan perubahaan struktur tabel yang telah dibuat seperti menambah field, menganti nama field, ataupun mengembalikan nama yang telah diubah. Perintah RENAME adalah perintah untuk mengubah nama tabel, perintah DROP digunakan untuk menghapus *database*, table, kolom ataupun index. Perintah terakhir yaitu SHOW perintah ini digunakan untuk menampilkan sebuah tabel yang ada.

Selain *Data Definition Language* (DDL), MySQL ada juga *Data Manipulation Language* (DML) bahasa ini digunakan untuk memanipulasi data dan mengelola data didalam database. DML ada empat perintah yaitu SELECT, INSERT, UPDATE, DELETE. Perintah SELECT ini adalah perintah untuk mengambil data dari suatu tabel. Perintah INSERT adalah perintah untuk memasukkan sebuah record baru ke dalam tabel database. Perintah UPDATE adalah perintah yang digunakan untuk mengubah data dan memodifikasi data yang terdapat dalam table. Perintah DELETE yang digunakan untuk menghapus data dalam record.

*Data Control Language* (DCL) adalah perintah dasar untuk query SQL yang berfungsi untuk mengatur hak yang dimiliki oleh *user*. DCL adalah perintah SQL yang berhubungan dengan manipulasi *user* dan hak akses privileges. Perintah DCL biasanya digunakan oleh admin untuk bisa menjaga keamanan dan kerahasiaan database. Ada 2 perintah yang digunakan dalam DCL yaitu GRANT dan REVOKE. Perintah GRANT adalah perintah untuk memberikan hak akses oleh admin ke *user* didatabase untuk bisa mengakses database tersebut. Admin bisa memberikan privillage ke *user* dari CREATE, SELECT, UPDATE, dan hak khusus yang berhubungan dengan sistem database. Perintah REVOKE adalah perintah untuk mencabut hak *user* dalam mengakses *database*.

### 2.2.4.3 Architecture My SQL




Gambar 2. 11 Architecture MySQL
Sumber: [46]

MySQL memiliki sebuah *architecture* bisa terlihat dari Gambar 2.11 yang memiliki beberapa lapisan. Lapisan pertama yang paling atas adalah *client*. Lapisan ini *client* karena *client* yang membuat sebuah instruksi dari ekspresi MySQL. Lapisan ini ada beberapa layanan yang penting yaitu keamanan karena ketika *client* berhasil terhubung ke server MySQL maka akan memeriksa *client* tersebut memiliki *privilleges* atau tidak dengan melihat autentikasi ketika *client* mengisikan *username* dan *password client* dan penangangan koneksi [24], [26].

Lapisan berikutnya yaitu server karena adalah otak dari My SQL yang termasuk kode, analisis, optimasi, caching, dan semua fungsi bawaan dari MySQL. Fungsi seperti prosedur yang tersimpan dan tampilan. Ada beberapa subkomponen dari server MySQL yaitu [24], [26]:

1. Thread Handling

   Sub komponen pertama yaitu *Thread Handling* terjadi saat klien mengirimkan permintaan tersebut ke *server* maka *server* akan menerima permintaan tersebut sehingga *client* bisa terhubung.

2. Parser

   Sub komponen *Parser* adalah jenis dari komponen *software* yang membangun struktur data dari input yang diberikan

3. Optimizer

   Ketika Sub komponen setelah selesai dalam penguraian maka ada berbagai jenis teknik pengoptimalan yang diterapkan di blok optimal. Teknik ini bisa dari menulis ulang query, memilih index dan urutan pemindaian tabel

4. Query Cache

   Sub komponen ini menyimpan set hasil lengkap untuk pernyataan query yang dimasukkan. Sebelum parsing maka MySQL akan berkonsultasi dengan komponen ini karena jika query yang di ketik oleh client identik maka server bisa melewatkan parsing, optimasi dan eksekusi sehingga hanya cukup menampilkan output dari *cache*

5. Buffer and Cache

   Sub komponen *Buffer and Cache* adalah menyimpan query atau masalah yang pernah terjadi. *Client* yang menulis query maka masuk ke query *cache* jika tidak ada masalah maka akan lanjut untuk memberikan output.

6. Table Metadata Cache

   Sub komponen *Table Metadata Cache* adalah area memori yang dicadangan untuk melacak suatu informasi dari database, *index*, dan object. Jika *database*, index, dan object berjumlah besar maka ukuran *cache* metadata juga besar.

7. Key Cache

   Sub komponen terakhir yaitu *key cache* yang memiliki peran unik untuk mengidentifikasi objek dalam *cache*.

Lapisan terakhir yaitu *storage engine*, lapisan ini bertanggung jawab untuk menyimpan dan mengambil semua data yang telah disimpan di *database*. Selain itu, layer ini juga dihitung sebagai RDBMS yang sering digunakan [24], [26].

### 2.2.5 Benchmark Database

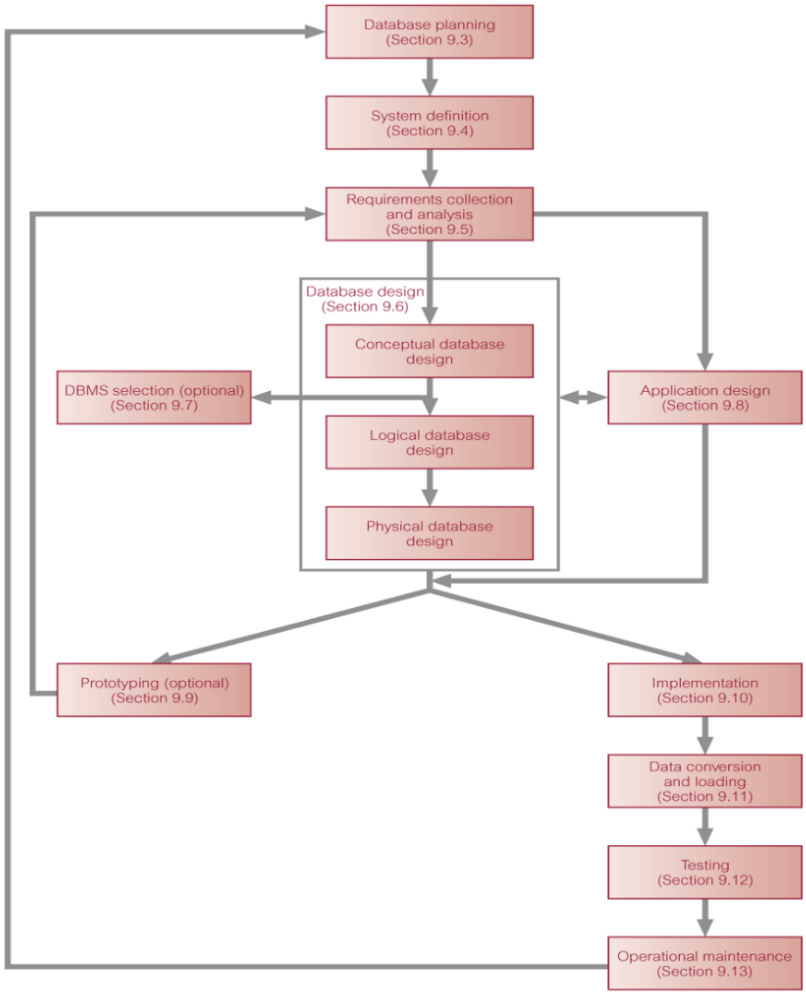
Dalam melakukan pengujian *performance database* maka diperlukan sebuah tolak ukur atau *benchmark* yang dapat digunakan. *Benchmark* dalam *database* terdapat benchmark untuk *NoSQL* dan RDBMS. Contoh dari *NoSQL* seperti *Yahoo! Cloud Serving Benchmark* (YCSB). YCSB merupakan rangkaian *benchmarking open-source* untuk menguji kinerja *Database Management System* (DBMS) yang *NoSQL*. Tujuan dari YCSB adalah memberikan sebuah standar pengukuran kinerja yang konsisten untuk *Database Management System* (DBMS) yang berbeda-beda. *Benchmark* YCSB mengukur dari *combination of client* yang menjalankan query, *workload generator* dirancang untuk mendukung dari *workload* yang mensimulasikan berbagai skenario yang telah dibuat, dan *core* [47]. YCSB sudah digunakan untuk membandingkan *relational database* dan *database NoSQL*. Namun, YCSB sendiri digunakan untuk *database NoSQL* seperti Cassandra, MongoDB dan Redis. YCSB dapat disesuaikan dengan model data *NoSQL* yang berbeda-beda [48].

Selain *benchmark NoSQL* yaitu YCSB terdapat juga *benchmark* RDBMS yaitu *Transaction Processing Performance Council* (TPC). TPC merupakan tolak ukur untuk RDBMS. TPC dibuat bersama dengan Microsoft, Intel, HP dan lainnya untuk dijadikan standar dalam mengevaluasi RDBMS. TPC menguji karakteristik ACID DBMS, *query speed*, dan kemampuan dalam *Online Transaction Processing* (OLTP). TPC ada beragam jenisnya. Namun dari beragam jenis TPC terdapat TPC-C dan TPC-E yang dirancang untuk *database Online Transaction Processing* (OLTP). Sementara untuk *benchmark Decision Support System* (DSS) maka menggunakan TPC-H dan TPC-DS [49].

TPC-C telah digunakan sejak tahun 1992 dan secara khusus digunakan untuk *Online Transaction Processing* (OLTP) seperti pemasukan dan pengiriman pesanan, pencatatan pembayaran, pengecekan status pesanan, dan pemantauan stok digudang. TPC tidak terbatas pada segmen bisnis tertentu. TPC-C cocok untuk *database* yang banyak tabel dengan berbagai macam ukuran, atribut dan relasi [50]. TPC-E merupakan *benchmark* untuk mengukur *Online Transaction Processing* (OLTP) yang baru dikembangan oleh TPC pada tahun 2006. TPC-E lebih kompleks dibandingkan *Online Transaction Processing* (OLTP) sebelumnya [51].

Selanjutnya *benchmark decision-system*, TPC-H merupakan tolak ukur yang mengukur sebuah keputusan sejak tahun 1999. TPC-H juga terdiri dari serangkaian *ad-hoc query* yang berorientasi pada bisnis dan modifikasi data secara bersamaan. TPC-H menggambarkan *Decision Support System* (DSS) yang memeriksa jumlah data sangat besar, mengeksekusi pertanyaan dengan kompleksitas tinggi dan perlu memberikan jawaban [52]. Selain ada TPC-H, terdapat juga TPC-DS sebagai *benchmark* untuk *decision support*. TPC-DS adalah *benchmark* yang memodelkan beberapa aspek sistem pendukung keputusan yang diterapkan secara umum termasuk dalam data *maintenance* dan query. TPC-DS menghasilkan pengukuran *response* query dalam mode single *user*, *throughput* query dalam mode *multi-user* dan data *maintenance* untuk *hardware*, *operation system*, dan konfigurasi sistem pemrosesan data tertentu seperti *multi-user decision support workload* [53].

## 2.3 Database System Development Lifecycle (DBLC)




Gambar 2. 12 Database System Development Life Cycle
Sumber: [54]

Gambar 2.12 merupakan *Database System Development Lifecycle* (DBLC). Menurut [38], dalam melakukan sebuah pengembangan pada sistem *database* terdapat sebuah metode yaitu menggunakan *Database System Development Lifecycle* (DBLC). Dalam pembuatan sistem database yang sederhana maka dapat membuat *Database System Development Lifecycle* (DBLC) yang lebih sederhana tapi ketika pembuatan sistem database yang kompleks maka dibutuhkannya *Database System Development Lifecycle* (DBLC) yang kompleks. *Database System Development Lifecycle* (DBLC) terdapat beberapa tahapan yang perlu dilakukan. Tahapan-tahapan yang terdapat di DBLC sebagai berikut [38]:

1. Database planning

   Tahap pertama yaitu *Database planning* yang merencanakan sebuah pengembangan sistem *database* dapat diwujudkan secara efisien dan seefektif mungkin. *Database planning* bisa dikatakan tahapan yang menjelaskan mengenai tujuan dari pembuatan *database* yang ingin dibuat seperti proses pengumpulan data, design dan format data. Pada tahapan ini juga perlu menentukan *mission statement* dan *mission objective*. *Mission Statement* dapat diartikan sebagai memperjelas tujuan utama dari penggunaan dan pembuatan *database* yang diperlukan secara efisien dan efektif. *Mission objectives* dapat diartikan bahwa *mission objectives* dapat mengidentifikasi tugas yang bisa dilakukan oleh *database*.

2. System Definition

   *System Definition* merupakan tahapan yang memperjelas ruang lingkup dan batasan dari sistem *database*. Oleh karena itu, sebelum membuat sebuah *database* maka diperlukan sebuah batasan-batasan sistem *database*. Batasan sistem *database* tidak hanya batasan dalam *user* tapi juga aplikasi.

3. Requirement Collection and Analysis

   *Requirement Collection and Analysis* merupakan proses pengumpulan dan melakukan analisis informasi mengenai perusahaan/organisasi yang ingin menggunakan *database*. Dalam melakukan pengumpulan informasi maka bisa melakukan dengan teknik *fact-finding*. Teknik tersebut bisa berupa wawancara, kuesioner, observasi, riset dan memeriksa dokumentasi. Dari pengumpulan informasi maka perlu dilakukan sebuah analisis untuk mengidentifikasi hal yang ingin dibutuhkan oleh perusahaan/organisasi dalam pembuatan database. Pada *requirement collection and analysis* terdapat 3 pendekatan yang dapat digunakan untuk mengelola *requirement database* yaitu *centralized approach*, *view integration approach*, ataupun kombinasi dari *view approach* dan *view integration*.

4. Database design

*Database design* merupakan tahapan untuk mendesain *database* yang akan mendukung dari *mission statement* dan *mission objective* perusahaan terhadap sistem *database*. Terdapat 2 pendekatan dalam *database design* yaitu *bottom-up* dan *top-down*. Pendekatan *bottom-up* adalah pendekatan yang sangat cocok untuk merancang *database* yang sederhana dengan jumlah atribut yang sedikit. Pendekatan ini tidak disarankan untuk rancangan *database* yang kompleks seperti jumlah atribut yang banyak karena akan sulit untuk menghubungkan semua atribut.

Oleh karena itu, ketika *database design* kompleks maka dapat menggunakan pendekatan *top-down*. Pendekatan *top-down* dapat digambarkan dengan konsep *Entity Relationship* (ER) dari identifikasi entitas dan hubungan setiap entitas. Terdapat beberapa pendekatan lainnya yaitu *inside-out* yang hampir sama dengan *bottom-up* tapi yang berbeda dengan *bottom up* yaitu terlebih dahulu mengidentifikasi entitas besar yang kemudian akan menyebar ke entitas lainnya yang berkaitan dengan entitas tersebut. Terakhir ada pendekatan *mixed strategy* yang mencampurkan *bottom-up* dan *top-down*. Dalam membuat database design terdapat 3 fase utama dari yaitu [55], [56]:

- Conceptual Database Design

  *Conceptual database design* merupakan tahapan pertama dalam desain database. *Conceptual Database* memiliki tujuan untuk membangun konseptual *database* yang mencerminkan kebutuhan bisnis dan memberikan penjelasan yang jelas mengenai entitas, *attributes*, *relationships*, dan mengenai batasan antar entitas. Proses dari *conceptual database design* yaitu:

  1. *Requirement Gathering* yang mengumpulkan kebutuhan bisnis dari perusahaan untuk dapat mengidentifikasi elemen data, relasi, dan batasan.

2. *Entity-relationship modelling* merupakan langkah untuk membuat sebuah model Enitity relationship (ER) yang mewakili entitas, atribut dan relasi

- Logical Database Design

  *Logical database design* memiliki tujuan untuk mengubah konseptual menjadi *logical structure database* seperti relasi. Dalam *logical database design* terdapat beberapa konsep *logical database* yang bisa digunakan seperti *strong entity types* berarti entitas yang tidak bergantung dengan entitas lain, *weak entity types* berarti entitas yang tergantung dengan *strong entity* dan *weak entity* tidak mempunyai *primary key* [57], *one-to-many*, *one-to-one*, *superclass/subclass relations types*, *many-to-many*, *complex relationship types*, *multi valued attributes*. Dalam *logical database design* maka perlu dilakukan pengujian kebenaran dengan teknik normalisasi. Dengan normalisasi ini memastikan bahwa relasi yang diturunkan tidak menyebabkan redundasi data dan anomaly.

- Physical Database Design

  *Physical Database Design* memiliki tujuan untuk memutuskan *logical structure* menjadi target dari *Database Management System* (DBMS) dan cara untuk *database* bisa diimplementasikan. *Physical database design* disesuaikan dengan sistem DBMS.

5. DBMS Selection

*DBMS Selection* merupakan tahapan untuk memilih *Database Management System* (DBMS) yang dijadikan pendukung dalam sistem database. Dalam memilih *Database Management System* (DBMS) maka ada beberapa tahapan untuk memilih *Database Management System* (DBMS) yaitu:

1. Define terms of reference study

   Tahapan ini berarti menetapkan tujuan dan ruang lingkup studi serta tugas yang perlu dilakukan berdasarkan kebutuhan

*user* yang akan menggunakan produk *Database Management System* (DBMS).

2. Shortlist two or three products

   Tahapan berikutnya menyediakan beberapa pilihan *Database Management System* (DBMS) yang ingin dievaluasi lebih lanjut. Beberapa pilihan *Database Management System* (DBMS) bisa sesuai juga dengan anggaran yang tersedia, kompatibilitas dengan *software* lain dan *hardware* serta kinerja dari *Database Management System* (DBMS) tersebut.

3. Evaluate products

   Langkah berikutnya melakukan evaluasi *product Database Management System* (DBMS) yang bisa dikelompokan berdasarkan *data definition*, *phyiscal definition*, *accesbility*, *transaction handling*, *utilities*, *development* dan fitur lainnya.

4. Recommend selection and produce report

   Langkah terakhir yaitu membuat sebuah dokumentasi proses evaluasi dan memberikan pernyataan temuan serta merekomendasi product *Database Management System* (DBMS).

*6. Application Design*

*Application Design* merupakan tahapan untuk perantara *user* dan *database*. Dalam *application design* terdapat 2 aspek yaitu *Transaction Design* dan *User Interface Design*. *Transaction Design* memiliki tujuan untuk mendefinisikan dan mendokumentasikan *high-level characteristics* dari transaksi yang dibutuhkan pada *database* agar *database* bisa mendukung semua transaksi yang diperlukan. Terdapat 3 transaksi utama yaitu *retrieval transaction* yang digunakan untuk mengambil data agar bisa ditampilkan dalam pembuatan report. Jenis transaksi kedua yaitu *update transactions* yang digunakan untuk menambahkan record baru, menghapus record lama dan mengubah record yang telah ada di *database*. Jenis transaksi terakhir yaitu *mixed transactions* yang dapat mengambil dan dapat mengubah datanya.

7. Prototyping

   *Prototyping* merupakan tahapan sebuah model dibuat untuk mengambarkan *database* yang ingin dibuat. Dengan tahapan *Prototyping* bertujuan untuk memberikan gambaran yang jelas dan biasanya ditemukan hal-hal yang ingin diperbaiki ataupun keinginan untuk dikembangkan.

8. Implementation

   *Implementation* merupakan tahapan untuk merealisasikan *database* sesuai dengan desain yang ada dengan menggunakan *Data Definition Language* (DDL) dari DBMS yang telah di pilih sebelumnya, dan bisa melakukan *Data Manipulation Language* (DML) dari DBMS yang telah dipilih sebelumnya.

9. Data conversion and Loading

   *Data conversion and loading* merupakan tahapan untuk memasukan data ke *database* yang telah dibuat. Jika terdapat data lama dengan format yang berbeda maka akan di konversi menjadi data dengan format yang baru.

10. Testing

    *Testing* merupakan tahapan untuk melihat kondisi sebuah *database* yang belum diluncurkan secara menyeluruh dengan kriteria penilaian *Learnability* yang berarti berapa lama waktu yang dibutuhkan *user* untuk menjalankan sistem ini, *Performance* seberapa cocok respon sistem ketika dijalankan, *Robustness* yang berarti seberapa toleran sistem terhadap kesalahan pengguna, *Recoverability* seberapa baik sistem memulihkan kesalahan *user*, dan *Adaptability*.

11. Operational Maintenance

    *Operational Maintenance* merupakan tahapan *monitoring* dan *maintenance* dari terhadap *database* yang telah dibuat dan memastikan berjalan dengan baik.

## 2.4 Tools

### 2.4.1 Database Management Tool

Dalam melakukan implementasi suatu *database* dengan *database* yang berbeda maka diperlukan sebuah tool yang mempermudah untuk menyimpan dan memproses data tersebut secara efisien. Dengan menggunakan *Database Management Tool* bisa membantu untuk mengelola kinerja aplikasi [58]. Ada beberapa jenis *Database Management Tool* sebagai berikut:

#### 2.4.1.1 DataGrip


Gambar 2. 13 Logo Datagrip
Sumber: [59]

Gambar 2.13 merupakan logo dari Datagrip. DataGrip merupakan *tool* berbayar yang dimiliki oleh Jetbrains untuk dapat melakukan query, create dan manage *database*. DataGrip mendukung juga *database* secara local, *cloud* atau *server*. DataGrip mendukung *database* seperti MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Redis, Azure dan *database* lainnya baik *relational* maupun *non-relational*. DataGrip dapat dijalankan di Linux, Windows, dan MacOs [60]. DataGrip dapat melakukan *export* dan *import* sesuai format teks seperti CSV, JSON, HTML. DataGrip juga bisa menyoroti perbedaan tabel satu sama lain. DataGrip menyediakan *Smart text editor*, *code completion* berarti DataGrip bisa menyelesaikan kode sehingga bisa mempercepat dalam menulis code, dan *Code analysis and quick fixed* berarti DataGrip bisa memberikan saran opsi terbaik dalam

penulisan SQL. DataGrip akan menyimpan *history* query dari query yang dijalankan [61], [62].

#### 2.4.1.2 Navicat


Gambar 2. 14 Logo Navicat Premium
Sumber: [63]

Gambar 2.14 merupakan logo dari Navicat Premium. Navicat merupakan *tool* yang telah ada sejak tahun 2002 yang bertujuan untuk menyederhanakan management di MySQL. Navicat juga *tool* yang berbayar ketika ingin menggunakan secara keseluruhan. Navicat memiliki beberapa fitur seperti Editor SQL Visual sehingga mempermudah dalam membuat query tanpa perlunya menghafal *syntax* SQL yang rumit. Navicat memiliki *user-interface* yang menarik dan mudah digunakan. Navicat menyediakan manajemen *database* lengkap seperti mengedit, menghapus, mengelola data dengan mudah. Selain itu, Navicat bisa migrasi data antar *database* yang berbeda platform dengan mudah dan aman. Navicat bisa dijalankan di Windows, Linux, dan MacOS. Navicat bisa terhubung dengan beberapa *database* seperti MySQL, Oracle, SQL Server, MariaDB, MongoDB dan beberapa *database* lainnya. Navicat juga bisa untuk *cloud database* seperti Amazon Aurora, Amazon Redshift, Oracle Cloud, Google Cloud. Navicat juga bisa melakukan *import* dan *export* dengan berbagai format seperti Excel, Access, CSV dan format lainnya [64], [65].

#### 2.4.1.3 DBeaver

Gambar 2. 15 Logo Dbeaver
Sumber: [66]

Gambar 2.15 merupakan logo dari DBeaver. DBeaver merupakan *tool* gratis yang mempermudah *user* untuk terhubung ke beberapa *database* seperti MySQL, PostgreSQL, Oracle, Microsoft SQL Server, SQLLite dan beberapa database *NoSQL, graph database*. DBeaver tersedia di Windows, Linux dan MacOS. DBeaver menjadi salah satu *tool* yang popular bagi *developer* dan *administrator database*. DBeaver ada beberapa fitur seperti bisa koneksi dengan beberapa *database* sehingga dalam *user-interface* yang sama bisa terhubung beberapa *database*. Fitur berikutnya yaitu editor SQL, *User interface* yang mudah dimengerti, DBeaver memiliki *export* dan *import* data dengan beberapa format file seperti CSV, Excel, SQL dump [66], [67].

### 2.4.2 Testing Tool

Dalam melakukan pengujian *performance database* maka ada beberapa *tools* yang dapat digunakan untuk melakukan pengujian tersebut. Hal ini dibutuhkan karena *tools* tersebut bisa menguji database secara komprehensif termasuk pengujian fungsional kinerja dan keamanan [68]. Terdapat beberapa *tools* yang bersifat *open-source* untuk melakukan pengujian *database* sebagai berikut:

#### 2.4.2.1 Apache JMeter


Gambar 2. 16 Logo JMeter
Sumber: [69]

Gambar 2.16 merupakan logo dari Apache JMeter. Apache JMeter merupakan *software* java *open-source* yang dirancang untuk memuat perilaku fungsional pengujian dan mengukur kinerja baik sumber daya statis dan dinamis, aplikasi dinamis web, ataupun mensimulasikan *workload* terhadap server, dan jaringan. JMeter *tools* sering digunakan untuk pengujian *database* dan *web*. Pada awalnya Apache JMeter digunakan untuk menguji aplikasi web tetapi *software* tersebut mengalami perkembangan untuk pengujian kinerja lainnya seperti layanan Web SOAP, FTP, *database* melalui *Java Database Connectivity* (JDBC), *Lightweight Directory Access Protocol* (LDAP), *Message-oriented middleware* (MO) via *Java Messaging Service* (JMS), Mail – SMTP(S), POP(S), IMAP(S), *Native commands* atau *shell scripts*, TCP/IP, dan *Java objects* [70].

Dalam pengujian *database performance* maka JMeter bisa menguji beberapa DBMS seperti MySQL, PostgreSQL, Oracle, SQL Server dengan bantuan sebuah *driver* yaitu *Java Database Connectivity*. Kelebihan dari JMeter yaitu dapat dijalankan dengan 2 cara yaitu GUI yang ramah bagi *user* atau no-GUI (*command-line interface),* mendukung untuk menguji *scalability* dan *load testing*, menyediakan hasil pelaporan dan analisis yang kuat karena dapat menganalisis *response-time*, identifikasi *bottlenecks*. Kelebihan berikutnya JMeter dapat lancar diberbagai platform seperti Window, MacOS, dan Linux. Namun JMeter memiliki kekurangan yaitu membutuhkan *memory* dan *processing power* untuk skala testing yang besar

[68], [71]. Ketika menjalankan JMeter maka perlu dilakukan beberapa step yaitu [72]:

1. Mempersiapkan data dan *environment* sesuai dengan konfigurasi komponen
2. Gunakan beberapa *sampler* untuk mensimulasikan permintaan pengguna
3. *Control the operation* dengan *thread group* untuk mengatur skenario operasi dan menggunakan pengontrol logika untuk mengontrol bisnis
4. Menyimpan hasil dengan *assertions* untuk menverifikasi hasil test dan menggunakan listeners untuk mengambil dan menampilkan hasil test.

JMeter hanya dapat dijalankan pada Java versi 8 keatas. Dalam melakukan pengujian performance database maka ada beberapa langkah yang dapat dilakukan sebagai berikut [72]:

1. Membuat rencana pengujian dan *thread group* merupakan langkah pertama dalam melakukan pengujian. *Thread group* menunjukkan jumlah pengguna yang disimulasikan oleh JMeter.
2. Menambahkan komponen konfigurasi yaitu *Java Database Connectivity connection* dengan cara menginstall terlebih dahulu driver JDBC sesuai DBMS yang diujikan. Setelah itu, bisa melakukan konfigurasi informasi koneksi database seperti *database* URL, kelas driver JDBC, nama pengguna *database*, *password*, nomor koneksi, dan atribut lainnya.
3. Tambah *sampler* JDBC pada elemen *thread group* dan konfigurasi informasi kumpulan koneksi seperti SQL *statement*, jenis eksekusi SQL, SQL parameter. Selain itu, ketika ingin melakukan test maka bisa menggunakan *query statement* dan *query condition* sesuai keinginan.

4. *Stress test debugging* merupakan langkah untuk menambahkan *result tree* agar dapat melihat hasil debug ketika *debug* selesai.
5. *Stress test running* merupakan langkah ketika sebelum menjalankan pengujian maka perlu menambahkan *aggregation report listener* yang ditambahkan di test plan element atau *thread group*. *Aggregation report* dapat memberikan ringkasan dari hasil pengujian kinerja.

Ketika melakukan pengujian *performance database* menggunakan JMeter maka hasil yang ditampilkan berupa *response time*, dan *throughput*. Namun JMeter juga dapat memantau beberapa indikator pemantauan seperti CPU Server, memori, IO disk selama melakukan pengujian berlangsung [72].

#### 2.4.2.2 HammerDB

Gambar 2. 17 Logo HammerDB
Sumber: [73]

Gambar 2.17 merupakan logo dari HammerDB. HammerDB merupakan *software benchmarking* gratis dan *open-source* yang dapat melakukan pengujian suatu *database* dengan *benchmark* TPC-C untuk *Online Transaction Processing* (OLTP) dan TPC-H untuk *Online Analytical Processing* (OLAP). *Tool* HammerDB dapat mendukung DBMS seperti Oracle, SQL Server, IBM DB2, MySQL, MariaDB, dan PostgreSQL [73]. HammerDB juga menyediakan beberapa dokumentasi untuk *user* dalam mencoba HammerDB sebagai *tool* pengujian. HammerDB dirancang untuk mendapatkan informasi mengenai hasil kinerja database baik di sisi *Online Transaction Processing* (OLTP) ataupun *Online Analytical Processing* (OLAP) [74]. HammerDB mempunyai *user interface* dan *command line forms*. HammerDB mendukung sistem Linux dan Windows. Terdapat

beberapa langkah untuk melakukan pengujian dengan HammerDB yaitu [72]:

1. Siapkan data pengujian, pada tahap ini menentukan *database* dan *benchmark* yang diujikan dengan memilih *options* pada *schema build* untuk mengonfigurasi informasi terkait *database*, lalu memilih *build* untuk membuat *database* pengujian *benchmark*.
2. *Configuration the test script* dengan memilih option di *driver scri*pt untuk mengonfigurasi informasi terkait *database*, durasi pengujian.
3. *Create vitual user and run the test*, pada langkah terakhir menentukan jumlah *virtual user* yang ingin diujikan dan telah bisa melakukan pengujian. Hasil dari pengujian ini dapat memantau *throughput* dalam transaction per minute

#### 2.4.2.3 SysBench

SysBench merupakan *tool open-source* yang melakukan pengujian kinerja *multi-thread* berdasarkan bahasa pemograman LuaJIT dengan *script* yang bisa disesuaikan. SysBench mencakup *benchmark* test dalam kinerja CPU, IO disk, memory, dan *database*. Pengujian menggunakan SysBench dapat diselesaikan dengan SysBench *benchmark* test *script*. SysBench mendukung *script* yang dicustomize untuk melakukan pengujian *perfomance*. SysBench hanya bisa dijalankan di Debian, Ubuntu, RHEL, CentOS, Fedora, dan macOS. SysBench telah tidak mendukung Windows sejak versi 1.0. SysBench lebih mendukung *database* MySQL dan PostgreSQL [72].

SysBench dapat melakukan pengujian *benchmark Online Transaction Processing* (OLTP) dan mensimulasikan *workload* dengan sistem pemrosesan transaksi yang sederhana. Dalam melakukan pengujian *benchmark Online Transaction Processing* (OLTP) dapat menggunakan skrip LUA yang disertakan dengan SysBench. Terdapat langkah-langkah pengujian dengan skrip LUA yang disertakan SysBench sebagai berikut [72]:

1. Persiapan data dengan membuat sebuah *database* untuk diuji. Dalam membuat *database* dapat menggunakan *command line* untuk membuat *table* dan data yang relevan dalam *database* yang ditentukan melalui skrip oltp.lua.
2. Menjalankan test dengan menggunakan command line dengan menentukan jumlah konkurensi untuk menguji table dan data yang telah dibuat. Hasil dari pengujian akan menunjukkan SQL statistic seperti queries performed, transactions, queries, ignored error. Selain itu ada general statistic, latency, dan thread fairness.
3. Setelah melakukan pengujian maka perlu clean data dengan command line untuk membersihkan tabel dan data yang telah diujikan.

### 2.4.3 Database Performance dan Pengukuran

Dalam *database Performance* ada beberapa faktor yang mempengaruhi dari performance database yaitu *Workload, Throughput, Resources, Optimization* dan *Contention*. Faktor pertama yaitu *Workload* adalah suatu kombinasi dari pengawasan pergudangan data, utilitas, perintah pengaturan yang dikelola di seluruh *Database Management System* (DBMS). Jika *workload* nya sangat besar maka ini mempengaruhi dengan kinerja *database*. Faktor berikutnya yaitu *Throughput* adalah kemampuan komputer untuk memproses data. *Throughput* adalah gabungan dari kecepatan I/O dan sistem operasi *software*. Faktor ketiga yaitu *Resources Software* adalah perangkat keras yang bisa mempengaruhi seperti memori, dan disk. Faktor keempat yaitu *Optimization Relational Database* yang dipengaruhi dari parameter *database*, parameter sistem. Faktor terakhir yaitu *Contention* [3].