BAB II
KAJIAN LITERATUR

2.1 Konsep *Concurrency*

Dalam sistem basis data yang digunakan oleh banyak pengguna, sering kali beberapa transaksi dijalankan pada waktu yang hampir bersamaan, terutama ketika banyak transaksi paralel mengakses dan memodifikasi data yang sama. Kondisi ini dikenal sebagai ***concurrency***. *Concurrency* memungkinkan sistem untuk mengeksekusi lebih dari satu transaksi secara bersamaan sehingga pemanfaatan sumber daya sistem menjadi lebih optimal [16].

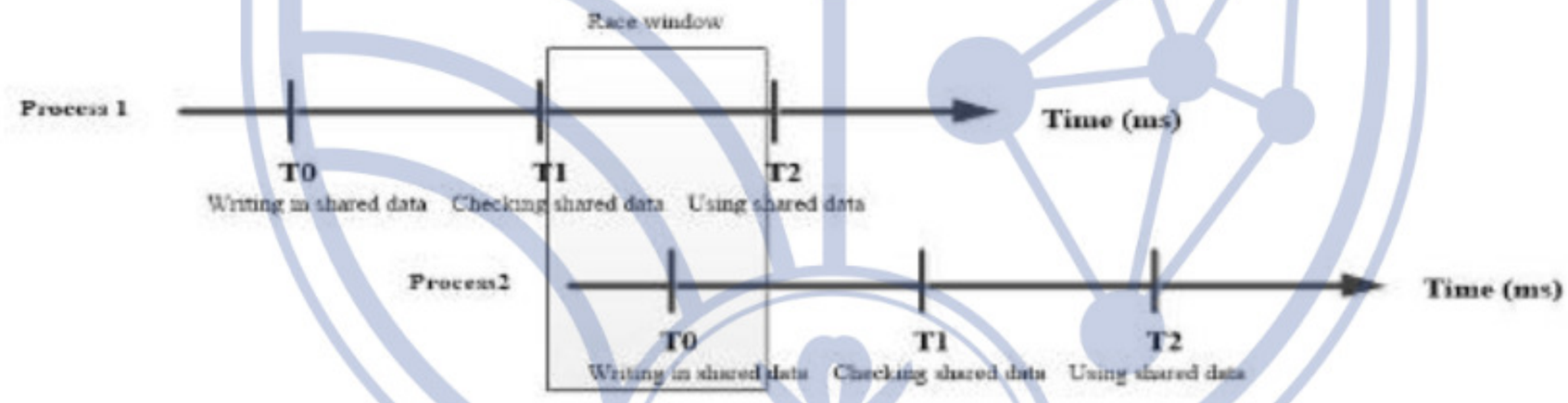
Pengelolaan transaksi yang berlangsung secara bersamaan tanpa mekanisme yang tepat dapat menimbulkan berbagai permasalahan. Kondisi tersebut dapat menyebabkan terjadinya konflik antar transaksi akibat akses data yang sama secara bersamaan. Dalam situasi tersebut, sistem perlu melakukan pembatalan (rollback) terhadap transaksi yang mengalami konflik agar transaksi lainnya dapat diselesaikan. Oleh karena itu, pengendalian *concurrency* diperlukan untuk memastikan bahwa konsistensi dan integritas data tetap terjaga selama proses eksekusi transaksi [16].

Dengan demikian, kegagalan dalam mengelola *concurrency* berpotensi menimbulkan kondisi *race condition*, yaitu situasi ketika dua atau lebih transaksi mengakses dan memodifikasi data yang sama pada waktu yang hampir bersamaan tanpa adanya kendali, sehingga hasil eksekusinya menjadi tidak sesuai dengan yang seharusnya. Kondisi ini dapat menyebabkan data yang tersimpan menjadi tidak konsisten dengan kondisi sebenarnya, sehingga informasi yang dihasilkan oleh sistem berpotensi tidak akurat. Oleh sebab itu, mekanisme *concurrency control* berperan penting dalam menjaga konsistensi dan integritas data ketika sistem menangani transaksi secara bersamaan..

2.2 *Race Condition*

Race condition merupakan salah satu masalah utama yang terjadi pada sistem yang bekerja secara konkuren. Masalah ini menjadi semakin krusial seiring dengan evolusi arsitektur perangkat keras modern yang mengandalkan pemrosesan paralel dan sistem *multi-core*. Perkembangan tersebut memungkinkan eksekusi ribuan instruksi secara bersamaan dalam satu waktu. Namun, kemampuan paralelisme ini juga memperbesar peluang terjadinya *race condition* apabila tidak diimbangi dengan mekanisme sinkronisasi yang tepat. Oleh karena itu, berbagai lapisan sistem komputer seperti aplikasi *multithreaded*, sistem operasi, komunikasi jaringan, sistem basis data, hingga *file systems* menjadi area yang sangat rentan terhadap *race condition* [17].

Masalah ini muncul ketika sumber daya bersama diakses oleh beberapa proses sekaligus tanpa mekanisme sinkronisasi yang memadai, di mana setidaknya satu proses melakukan perubahan pada data tersebut [18]. Salah satu contoh umum dari race condition adalah *Time-of-Check* to *Time-of-Use* (TOCTOU), yaitu kondisi di mana status suatu *resource* sudah berubah antara saat pemeriksaan dilakukan dan saat *resource* tersebut digunakan. Jeda waktu antara kedua momen ini menciptakan *race window*, yaitu periode singkat di mana resource berada dalam kondisi tidak konsisten dan dapat dieksploitasi [19]. Ilustrasi mengenai mekanisme terjadinya *race window* ini dapat dilihat pada Gambar 2.1.



Gambar 2. 1 Mekanisme Ilustrasi Terjadinya *Race Condition [18]*

Race condition dapat menyebabkan masalah serius pada stabilitas sistem dan konsistensi data. Kegagalan dalam menangani *race condition* bahkan dapat berujung fatal, seperti pada kasus mesin *Therac-25* yang menyebabkan pasien menerima dosis radiasi jauh lebih tinggi dari yang seharusnya [17]. Insiden tersebut menunjukkan bahwa masalah *race condition* dapat berakibat langsung pada keselamatan manusia. Selain dampak fisik, *race condition* juga dapat menimbulkan kerugian finansial dan reputasi bagi perusahaan. Selain itu, *race condition* juga dapat menimbulkan kerugian finansial dan reputasi bagi perusahaan [8]. Sebagai contoh, gangguan layanan *Amazon DynamoDB* pada *AWS* pada Oktober 2025 dipicu oleh *race condition* pada sistem otomasi *DNS*, di mana pengecekan validitas plan

menjadi usang akibat keterlambatan pemrosesan, sehingga seluruh catatan *DNS endpoint* regional terhapus dan memicu gangguan berantai pada ribuan platform pihak ketiga [20]. *CyberCube* memperkirakan kerugian ekonomi dari insiden ini mencapai 38 juta hingga 581 juta dolar AS, dengan dampak terhadap hampir 70.000 organisasi [21].

2.3 Algoritma *Concurrency Control*

Dalam pengembangan sistem basis data, algoritma untuk menangani akses data secara bersamaan sangatlah penting. Sistem Manajemen Basis Data (DBMS) menyediakan mekanisme untuk memastikan bahwa operasi yang dilakukan oleh banyak pengguna sekaligus tidak menyebabkan kerusakan data. Keandalan sebuah sistem informasi sangat bergantung pada kemampuannya menjaga integritas data ketika banyak transaksi dijalankan secara bersamaan. Tanpa adanya pengaturan yang terstruktur, tumpang tindih perintah baca dan tulis dapat memicu anomali yang merusak validitas informasi di dalam basis data. Untuk mencegah hal tersebut, algoritma yang umum diterapkan adalah mekanisme *concurrency control*.

Concurrency control merupakan sebuah mekanisme terstruktur yang bertujuan untuk menjaga konsistensi eksekusi berbagai transaksi di dalam sistem. Mekanisme ini bekerja dengan memastikan bahwa setiap transaksi tidak akan saling memengaruhi satu sama lain meskipun dijalankan secara bersamaan [15], [22]. Algoritma ini mengambil alih pengaturan lalu lintas data tersebut untuk memastikan bahwa setiap perintah diproses dengan pemantauan yang sangat ketat. Dengan adanya pengendalian yang sistematis ini, sistem dapat terhindar dari potensi tumpang tindih perintah yang dapat menghasilkan *output* yang keliru.

Tujuan paling mendasar dari implementasi *concurrency control* adalah menjamin bahwa transaksi-transaksi yang berjalan beriringan tersebut tetap dapat mempertahankan konsistensi serta integritas data secara utuh [12]. Algoritma ini dirancang secara khusus untuk menciptakan suatu skema eksekusi di mana setiap transaksi paralel diproses seolah-olah berjalan secara berurutan atau serial. Pendekatan eksekusi yang menyerupai proses serial ini sangat penting untuk mencegah adanya perubahan data yang hilang, tertimpa, atau tidak terbaca oleh transaksi lain. Melalui mekanisme perlindungan operasional ini, validitas informasi di dalam basis data akan selalu terjaga kemurniannya meskipun diakses oleh berbagai pihak secara serentak. Pada akhirnya, keberhasilan penerapan algoritma *concurrency control* akan sangat menentukan tingkat stabilitas operasional dari sistem manajemen basis data secara keseluruhan.

2.3.1 Optimistic Concurrency Control

Optimistic Concurrency Control (OCC) merupakan salah satu pendekatan *concurrency control* yang mengasumsikan bahwa konflik antar transaksi jarang terjadi. OCC mengizinkan transaksi untuk mengakses dan memodifikasi data tanpa memperoleh lock terlebih dahulu. Setiap transaksi bekerja secara independen pada salinan data lokal tanpa memblokir transaksi lain yang mengakses data yang sama. Validasi terhadap konflik baru dilakukan pada saat transaksi akan di-*commit*, bukan pada saat data diakses. Pendekatan ini merupakan kebalikan dari PCC yang mencegah konflik sejak awal melalui mekanisme *locking*.

Mekanisme kerja OCC terdiri dari tiga fase yang dieksekusi secara berurutan, yaitu fase *read*, fase *validation*, dan fase *write* [23]. Pada fase read, transaksi membaca data dari basis data dan melakukan seluruh operasi komputasi pada ruang kerja lokal tanpa memengaruhi data asli. Pada fase *validation*, sistem memeriksa apakah transaksi tersebut mengalami konflik dengan transaksi lain yang telah di-*commit* selama fase read berlangsung. Apabila tidak ditemukan konflik, transaksi memasuki fase *write* dan seluruh perubahan ditulis ke basis data secara permanen. Namun apabila konflik terdeteksi pada fase validation, transaksi dibatalkan dan harus diulang dari awal.

OCC cocok digunakan pada sistem dengan tingkat konflik yang rendah, di mana mayoritas transaksi mengakses data yang berbeda sehingga jarang terjadi konflik pada fase validation. Namun pada sistem dengan tingkat konflik yang tinggi, banyak transaksi yang akan gagal pada fase *validation* karena konflik baru terdeteksi setelah seluruh operasi pada fase *read* selesai dilakukan. Oleh karena itu, kesesuaian OCC sangat bergantung pada karakteristik konflik dari sistem yang akan dibangun.

2.3.2 *Pessimistic Concurrency Control*

Pessimistic Concurrency Control (PCC) merupakan salah satu pendekatan dalam *concurrency control* yang paling sering diimplementasikan pada sistem manajemen basis data. Sesuai dengan penamaannya, algoritma ini dibangun di atas sebuah asumsi pesimis bahwa konflik antar transaksi mungkin terjadi. Oleh karena itu, PCC selalu mengasumsikan adanya kemungkinan penggunaan atau modifikasi entitas data yang sama secara bersamaan oleh berbagai pengguna dalam suatu waktu. Dari asumsi kerentanan tersebut, algoritma ini mengambil langkah pencegahan sejak awal sebelum sebuah transaksi mulai melakukan operasi pada data. Pendekatan preventif ini dinilai lebih sesuai untuk sistem dengan tingkat konflik tinggi.

Dalam praktiknya, mekanisme dasar yang diandalkan oleh algoritma PCC adalah penerapan *locking* pada data yang sedang diproses oleh suatu transaksi aktif. Algoritma ini mengharuskan setiap transaksi untuk memperoleh *lock* terlebih dahulu sebelum melakukan operasi pada data yang berpotensi konflik dengan transaksi lain. Ketika sebuah entitas data telah di-*lock* oleh suatu transaksi, transaksi lain yang mencoba memperoleh *lock* yang konflik pada entitas tersebut harus menunda operasinya dan masuk ke dalam antrean. Transaksi yang tertunda ini baru akan diizinkan untuk melanjutkan eksekusinya setelah transaksi pemegang *lock* melepaskan *lock* yang dipegangnya [24].

Meskipun mekanisme *locking* efektif dalam menjaga konsistensi data, pendekatan ini mempunyai kelemahan berupa potensi terjadinya *deadlock*, yaitu kondisi ketika dua atau lebih transaksi saling menunggu pelepasan *lock* yang dipegang oleh transaksi lain sehingga tidak ada satu pun transaksi yang dapat melanjutkan eksekusinya. Kondisi seperti ini menyebabkan kedua transaksi tidak dapat melanjutkan prosesnya tanpa adanya intervensi dari sistem [25]. Salah satu pendekatan yang dapat digunakan untuk menangani *deadlock* adalah *deadlock detection*, di mana sistem memeriksa keberadaan siklus pada hubungan ketergantungan antar transaksi untuk mengidentifikasi terjadinya *deadlock*, lalu membatalkan salah satu transaksi yang terlibat agar *lock* yang dipegangnya dapat dilepaskan [26].

Penerapan algoritma *Pessimistic Concurrency Control* ini memberikan jaminan konsistensi serta integritas data pada lingkungan basis data dengan tingkat konflik yang tinggi. Melalui mekanisme locking, sistem dapat mencegah terjadinya anomali yang disebabkan oleh akses konkuren terhadap data yang sgama. Pendekatan yang mengutamakan pencegahan ini memastikan bahwa setiap transaksi yang berjalan tidak akan tumpang tindih dengan transaksi lain pada data yang sama. Salah satu protokol PCC yang diterapkan secara terstruktur untuk mengimplementasikan konsep penguncian pesimis ini adalah *Two-Phase Locking* (2PL).

2.3.2.1 Two-Phase Locking (2PL)

Salah satu protokol berbasis *locking* yang paling banyak diandalkan dalam implementasi *Pessimistic Concurrency Control* (PCC) adalah *Two-Phase Locking* (2PL) [24]. Protokol 2PL ini didesain secara khusus untuk mengoordinasikan akses ke berbagai sumber daya yang dibagikan (*shared resources*) dengan menggunakan mekanisme penguncian yang terstruktur. Tujuan utama dari koordinasi akses tersebut adalah untuk memastikan bahwa urutan eksekusi akhir dari berbagai transaksi paralel akan selalu meniru

urutan eksekusi secara serial [27]. Dengan meniru urutan eksekusi serial ini, sistem basis data dapat menjamin tingkat *serializability* yang tinggi guna mencegah terjadinya konflik antar transaksi. Dengan demikian, 2PL menjamin bahwa hasil eksekusi transaksi-transaksi yang berjalan secara konkuren setara dengan hasil eksekusi serial, sehingga konsistensi dan integritas data dapat terjaga."

Sesuai dengan penamaannya, mekanisme operasional dari protokol ini membagi jalannya eksekusi sebuah transaksi ke dalam dua fase utama yang saling berbeda secara tegas [28]. Fase pertama dikenal luas dengan sebutan *Growing Phase*, di mana sebuah transaksi diperbolehkan untuk meminta dan memperoleh *locks* baru yang dibutuhkannya. Namun, selama masa *Growing Phase* ini berlangsung, transaksi tersebut sama sekali tidak diizinkan untuk melepaskan satu pun *lock* yang telah berhasil dikuasainya. Setelah transaksi memutuskan untuk melepaskan setidaknya satu *lock*, proses tersebut akan langsung memasuki fase kedua yang disebut sebagai *Shrinking Phase*. Pada tahapan *Shrinking Phase* ini, transaksi hanya diperbolehkan untuk melepaskan *locks* yang dipegangnya secara bertahap dan dilarang keras untuk kembali memperoleh *lock* baru dalam bentuk apa pun.

Mekanisme kerja protokol 2PL ini dapat direpresentasikan dengan jelas melalui penerapan skema penguncian pada Tabel 2.1 yang secara ketat menjamin *serializability* data. Dalam ilustrasi mekanisme tersebut, transaksi T1 diwajibkan melewati *growing phase* dan secara spesifik menahan seluruh *lock* nya secara eksklusif hingga mencapai tahapan *Commit* dan *Unlock*. Ketatnya aturan penguncian ini memastikan bahwa data yang sedang dimodifikasi oleh transaksi T1 akan tetap terisolasi dengan aman hingga semua perubahannya bersifat permanen. Sementara itu, guna mencegah terjadinya inkonsistensi data, sistem menerapkan mekanisme penundaan pada transaksi T2. Ketika T2 mengajukan permintaan *shared lock* melalui perintah S-*LOCK*(A) di saat data telah di *lock* dan masih dimodifikasi oleh transaksi T1, permintaan tersebut tidak langsung dikabulkan (*granted*). Transaksi T2 baru diizinkan untuk melakukan operasi pembacaan R(A) setelah transaksi T1 membebaskan X-*LOCK* (X) nya pada data tersebut.

Tabel 2. 1 Mekanisme 2PL

T1	T2
X-*LOCK*(A)	
R(A)	S-*LOCK*(A)
A=A-100	
W(A)	
X-*LOCK*(B)	
R(B)	

B=B+100	
W(B)	
COMMIT	
UNLOCK(A)	
UNLOCK(B)	
	R(A)
	S-LOCK(B)
	R(B)
	ECHO A+B
	COMMIT
	UNLOCK(A)
	UNLOCK(B)

2.3.2.2 Bamboo

Bamboo merupakan algoritma *concurrency control* yang dirancang untuk mengurangi kontensi pada *hotspot* dalam sistem basis data [28]. *Hotspot* didefinisikan sebagai sejumlah kecil baris data yang diakses oleh banyak transaksi konkuren secara bersamaan, yang menjadi sumber utama penurunan kinerja pada *workload* transaksional dengan kontensi tinggi. Algoritma ini dirancang berdasarkan pengamatan bahwa pada protokol 2PL konvensional, sebuah *lock* baru dilepas setelah seluruh transaksi *commit* meskipun bagian *hotspot* mungkin hanya menempati sebagian kecil dari waktu eksekusi transaksi. *Bamboo* mengusulkan pelanggaran terkendali terhadap aturan kedua 2PL dengan membolehkan pelepasan *lock* lebih awal pada titik tertentu dalam transaksi.

Cara kerja *Bamboo* berpusat pada konsep *lock retire*, yaitu pemindahan sebuah *lock* dari status aktif menjadi status retired segera setelah operasi tulis terakhir pada suatu *tuple* selesai [28]. Setelah *lock* di-*retire*, transaksi lain diperbolehkan membaca data *uncommitted* (*dirty read*) dari transaksi pemilik *lock* tersebut tanpa harus menunggu *commit* selesai. Untuk mendukung mekanisme ini, *Bamboo* menambahkan struktur data berupa daftar *retired* pada setiap *lock entry serta* variabel *commit_semaphore* pada setiap transaksi guna melacak *dependency* antar transaksi. Sebelum sebuah transaksi diperbolehkan *commit*, nilai *commit_semaphore*-nya harus mencapai nol, yang menandakan seluruh transaksi yang menjadi dependensinya telah selesai terlebih dahulu.

Meskipun menawarkan peningkatan throughput pada *workload* dengan kontensi tinggi, *Bamboo* memiliki sejumlah kelemahan yang perlu dipertimbangkan [28]. Kelemahan utama terletak pada munculnya *cascading abort* sebagai konsekuensi langsung dari diizinkannya pembacaan data *uncommitted*, apabila sebuah transaksi yang telah *retire lock* kemudian *abort*, maka seluruh transaksi yang telah membaca data *dirty*-nya juga harus

dibatalkan secara berantai. Selain itu, *Bamboo* hanya memberikan peningkatan kinerja yang nyata apabila akses ke *hotspot* merupakan bagian kecil dari keseluruhan eksekusi transaksi, sehingga sisa operasi dalam transaksi tersebut masih dapat dijalankan secara paralel oleh transaksi lain.

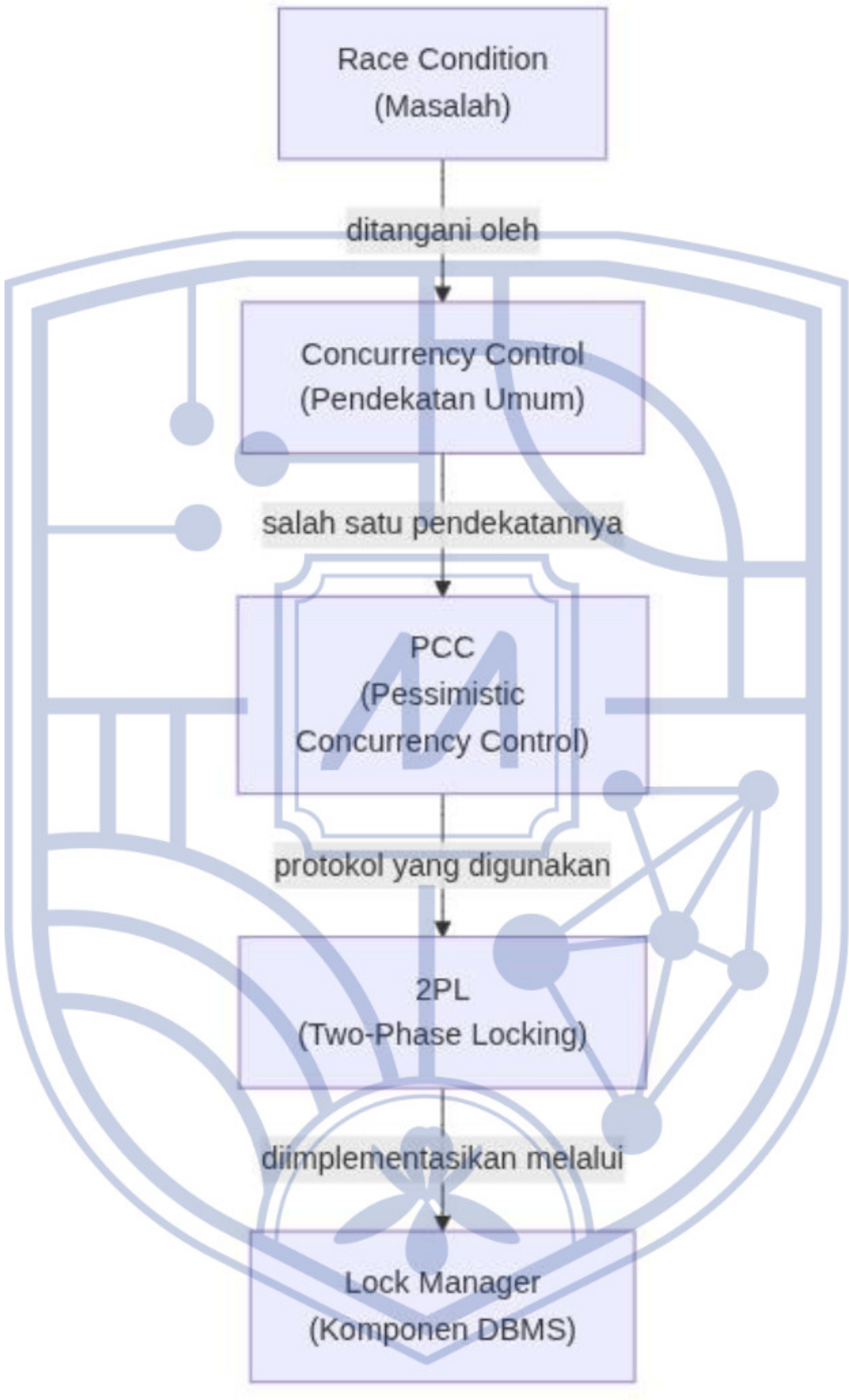
2.3.3 Lock Manager

Lock manager merupakan komponen dalam DBMS yang berfungsi untuk menangani permintaan penguncian (*lock* request) dari berbagai transaksi serta memberikan hak akses terhadap data yang diperlukan. Komponen ini berperan sebagai pusat pengelolaan *lock* yang bertugas menentukan apakah suatu transaksi dapat memperoleh *lock* atau harus menunggu hingga *lock* dilepaskan oleh transaksi lain. Sebelum suatu transaksi mengakses objek data tertentu, transaksi tersebut harus terlebih dahulu memperoleh *lock* dari *lock manager*. Setelah transaksi selesai, *lock* tersebut akan dilepaskan kembali [29]. Dengan mekanisme ini, sistem dapat mengatur akses data secara bersamaan, menjaga konsistensi data, serta mencegah terjadinya *race condition*.

Sebagai implementasi dari mekanisme *lock manager* pada sistem manajemen basis data, InnoDB pada MySQL menerapkan mekanisme *locking* pada tingkat baris data (row-level *locking*). Mekanisme *locking* ini digunakan untuk mengatur akses objek data oleh beberapa transaksi yang berjalan secara bersamaan. Ketika suatu transaksi mengakses atau melakukan perubahan pada objek data dalam tabel, sistem terlebih dahulu memeriksa apakah baris data yang akan diakses sedang dipegang lock-nya oleh transaksi lain dengan mode yang konflik atau tidak Jika baris data tersebut tidak sedang di lock, maka transaksi dapat langsung memperoleh *lock* dan melanjutkan proses akses data. Namun apabila baris data tersebut sedang di lock oleh transaksi lain, maka transaksi yang baru harus menunggu hingga *lock* tersebut dilepaskan sebelum dapat mengakses data tersebut [30].

Dalam mekanisme *locking* yang diterapkan pada InnoDB terdapat beberapa mode penguncian (*lock* modes) yang digunakan untuk mengatur jenis akses terhadap objek data. Dua mode penguncian dasar yang digunakan adalah *Shared Lock* (S) dan *Exclusive Lock* (X). *Shared lock* memungkinkan suatu transaksi membaca baris data tanpa menghalangi transaksi lain untuk melakukan pembacaan pada baris yang sama, namun menghalangi transaksi lain yang ingin memperoleh exclusive lock pada baris tersebut [30]. Sebaliknya, *Exclusive lock* diberikan ketika suatu transaksi membutuhkan akses eksklusif terhadap baris data, seperti pada operasi pembaruan, penghapusan, maupun operasi baca yang menggunakan pernyataan SELECT FOR UPDATE, sehingga transaksi lain tidak dapat

mengakses baris data tersebut hingga lock dilepaskan. [30]. Dengan penerapan mode penguncian ini, sistem basis data dapat menghindari konflik antar transaksi, mencegah terjadinya *race condition*, serta menjaga konsistensi data yang diakses secara bersamaan.



Gambar 2. 2 Hubungan Antar Konsep *Race Condition*, *Concurrency Control*, PCC, 2PL, Dan *Lock Manager*

2.3.4 Lock Wait

Dalam arsitektur sistem manajemen basis data MySQL, storage engine InnoDB memiliki mekanisme internal yang sangat terstruktur untuk mengelola informasi *lock* dan *lock-wait*. Ketika sebuah transaksi melakukan update pada suatu baris data atau secara

eksplisit menguncinya menggunakan perintah *SELECT FOR UPDATE*, InnoDB secara otomatis akan membentuk sebuah daftar atau antrean penguncian pada baris tersebut. Mekanisme pembentukan antrean yang serupa juga diterapkan oleh sistem ketika InnoDB perlu mempertahankan daftar penguncian pada tingkat tabel. Apabila terdapat transaksi kedua yang mencoba memperbarui baris atau mengunci tabel yang telah di *lock* sebelumnya oleh transaksi lain dalam mode yang tidak kompatibel, maka sistem harus mencegah terjadinya tumpang tindih operasi. Sebagai bentuk respons pencegahan, InnoDB akan menambahkan *lock* request dari transaksi baru tersebut ke dalam antrean *lock* yang bersesuaian agar integritas data tetap terjaga [31].

Agar sebuah *lock* data dapat sepenuhnya dikuasai oleh suatu transaksi yang sedang mengantre, seluruh permintaan *lock* tidak kompatibel yang telah masuk lebih dulu ke dalam antrean harus disingkirkan terlebih dahulu. Proses penyingkiran *lock* dari antrean ini umumnya dieksekusi secara otomatis ketika transaksi terdahulu yang memegang atau meminta *lock* tersebut telah menyelesaikan operasinya melalui tahapan *commit* atau *roll back*. Sepanjang siklus hidupnya, sebuah transaksi tunggal sejatinya dapat memiliki jumlah permintaan penguncian yang tidak terbatas untuk berbagai baris maupun tabel yang berbeda di dalam basis data. Pada suatu waktu tertentu, transaksi tersebut mungkin meminta *lock* yang sedang dipertahankan oleh transaksi lain, sehingga status eksekusinya menjadi terblokir dan harus menunggu transaksi pemblokir tersebut selesai. Berdasarkan kondisi operasional tersebut, InnoDB mengklasifikasikan transaksi menjadi berstatus *RUNNING* jika tidak sedang menunggu *lock*, dan berstatus *LOCK WAIT* apabila transaksi tertahan dalam antrean, di mana nilai status ini dapat dipantau melalui tabel INNODB_TRX pada *INFORMATION_SCHEMA* [31] .

Untuk memfasilitasi pemantauan dan analisis kelancaran transaksi, MySQL menyediakan arsitektur Performance Schema yang menyimpan rincian kendala penguncian secara komprehensif. Di dalam skema tersebut, terdapat tabel *data_lock*s yang menyimpan representasi baris informasi mengenai status penguncian transaksi di dalam sistem, sebagaimana yang dapat dilihat pada Gambar 2.2. Melalui ilustrasi pada gambar tersebut, tabel pemantauan ini secara spesifik menampilkan berbagai rincian parameter teknis seperti *LOCK_TYPE*, *LOCK_MODE*, hingga *LOCK_STATUS*. Informasi yang disajikan tersebut juga mendeskripsikan secara rinci masing-masing objek yang di*lock*, baik pada tingkat tabel maupun pada tingkat baris data. Sebagai pelengkap dalam melacak rantai kebuntuan operasi, tabel data_*lock_wait*s turut disediakan di dalam skema untuk memperlihatkan secara

eksplisit *locks* mana yang sedang dipegang oleh suatu transaksi dan bertindak sebagai pemblokir bagi transaksi lainnya [32].

```
mysql> SELECT * FROM performance_schema.data_locks\G
*************************** 1. row ***************************
               ENGINE: INNODB
       ENGINE_LOCK_ID: 139664434886512:1059:139664350547912
ENGINE_TRANSACTION_ID: 2569
            THREAD_ID: 46
             EVENT_ID: 12
        OBJECT_SCHEMA: test
          OBJECT_NAME: t1
       PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
           INDEX_NAME: NULL
OBJECT_INSTANCE_BEGIN: 139664350547912
            LOCK_TYPE: TABLE
            LOCK_MODE: IX
          LOCK_STATUS: GRANTED
            LOCK_DATA: NULL
*************************** 2. row ***************************
               ENGINE: INNODB
       ENGINE_LOCK_ID: 139664434886512:2:4:1:139664350544872
ENGINE_TRANSACTION_ID: 2569
            THREAD_ID: 46
             EVENT_ID: 12
        OBJECT_SCHEMA: test
          OBJECT_NAME: t1
       PARTITION_NAME: NULL
    SUBPARTITION_NAME: NULL
           INDEX_NAME: GEN_CLUST_INDEX
OBJECT_INSTANCE_BEGIN: 139664350544872
            LOCK_TYPE: RECORD
            LOCK_MODE: X
          LOCK_STATUS: GRANTED
            LOCK_DATA: supremum pseudo-record
```

Gambar 2. 3 Detail Informasi *Locks* Melalui *Performance Schema*

2.4 *Race Condition Testing*

Pengujian *race condition* pada aplikasi web tidak dapat dilakukan hanya dengan mengirimkan beberapa *request* secara bersamaan secara sederhana. Hal ini disebabkan oleh adanya *race window*, yaitu rentang waktu dimana beberapa *thread* mengakses data yang sama [33] . Agar *race condition* terjadi, beberapa *request* harus dapat masuk dan diproses dalam rentang waktu tersebut. Namun terdapat beberapa faktor yang mempengaruhi *race window*

Salah satu faktor yang mempersulit pengujian *race condition* adalah *network jitter*, yaitu variasi waktu antara pengiriman dan penerimaan *request* melalui jaringan [34]. Variasi ini dapat menyebabkan *request* yang dikirim secara bersamaan oleh *client* tiba di *server* pada waktu yang berbeda-beda. Akibatnya, kemungkinan beberapa *request* berada dalam *race window* yang sama menjadi lebih kecil, sehingga pengujian *race condition* menjadi kurang efektif apabila hanya mengandalkan pengiriman *request* paralel biasa.

Salah satu aplikasi yang dapat digunakan untuk membantu pengujian *race condition* adalah *Burp Suite* yang dipadukan dengan ekstensi *Turbo Intruder*. *Turbo Intruder*

menyediakan kemampuan untuk mengirimkan *request* dalam jumlah besar dengan kontrol yang lebih presisi terhadap waktu pengiriman melalui teknik *single-packet attack*. Teknik ini memanfaatkan *HTTP/2* dengan membuka beberapa *stream* secara bersamaan, di mana pengiriman *frame* terakhir dari setiap *stream* ditahan terlebih dahulu, kemudian seluruh *frame* terakhir tersebut dikirimkan sekaligus dalam satu paket TCP [35]. Karena server *HTTP/2* baru memproses sebuah *request* setelah seluruh *stream*-nya diterima secara lengkap, cara ini memastikan semua *request* mulai diproses pada waktu yang hampir identik di sisi *server*, sehingga kemungkinan beberapa request masuk ke dalam *race window* yang sama menjadi lebih besar.

Jumlah request yang dikirim juga menjadi pertimbangan dalam pengujian race condition. Beberapa penelitian menggunakan batch berukuran 20 request pada pengujian berbasis single-packet attack dan berhasil membuktikan terjadinya race condition [19]. Berdasarkan hal tersebut, nilai 20 request dapat dijadikan acuan sebagai beban dasar dalam pengujian race condition. Jumlah ini selanjutnya dapat ditingkatkan untuk menguji konsistensi perilaku sistem pada beban konkuren yang lebih besar.

2.5 *Black Box Testing*

Evaluasi fungsionalitas aplikasi menjadi tahapan krusial dalam siklus hidup pengembangan perangkat lunak untuk menjamin kualitas produk akhir. Salah satu pendekatan yang umum digunakan untuk melakukan evaluasi tersebut adalah metode *black box testing*. *Black box testing* berfokus pada penilaian fungsionalitas sistem berdasarkan spesifikasi yang ada tanpa memedulikan implementasi kode sumber di baliknya [36]. Melalui metode ini, analisis dilakukan secara dinamis dengan memberikan data masukan serta memeriksa luaran yang dihasilkan sistem. Proses pengujian ini diaplikasikan pada setiap fungsionalitas aplikasi guna memvalidasi kesesuaian program dengan spesifikasi desain awal [37].

Keuntungan utama dari pengujian *black box* adalah prosesnya dapat dilakukan oleh pihak yang tidak menguasai pengetahuan teknis seputar pemrograman perangkat lunak. Metode ini dimanfaatkan untuk mengidentifikasi berbagai kelemahan atau mencari kondisi spesifik yang dapat menyebabkan sistem gagal beroperasi. Penelusuran kelemahan tersebut dilakukan demi memvalidasi dan menjamin bahwa produk akhir yang dihasilkan telah sesuai sepenuhnya dengan rancangan spesifikasi awal [38]. Ada beberapa teknik yang dapat digunakan dalam *black box testing*, seperti *equivalence partitioning*, *boundary value analysis*, *decision table testing*, *graph-based testing*, *state transition testing*, *use-case*

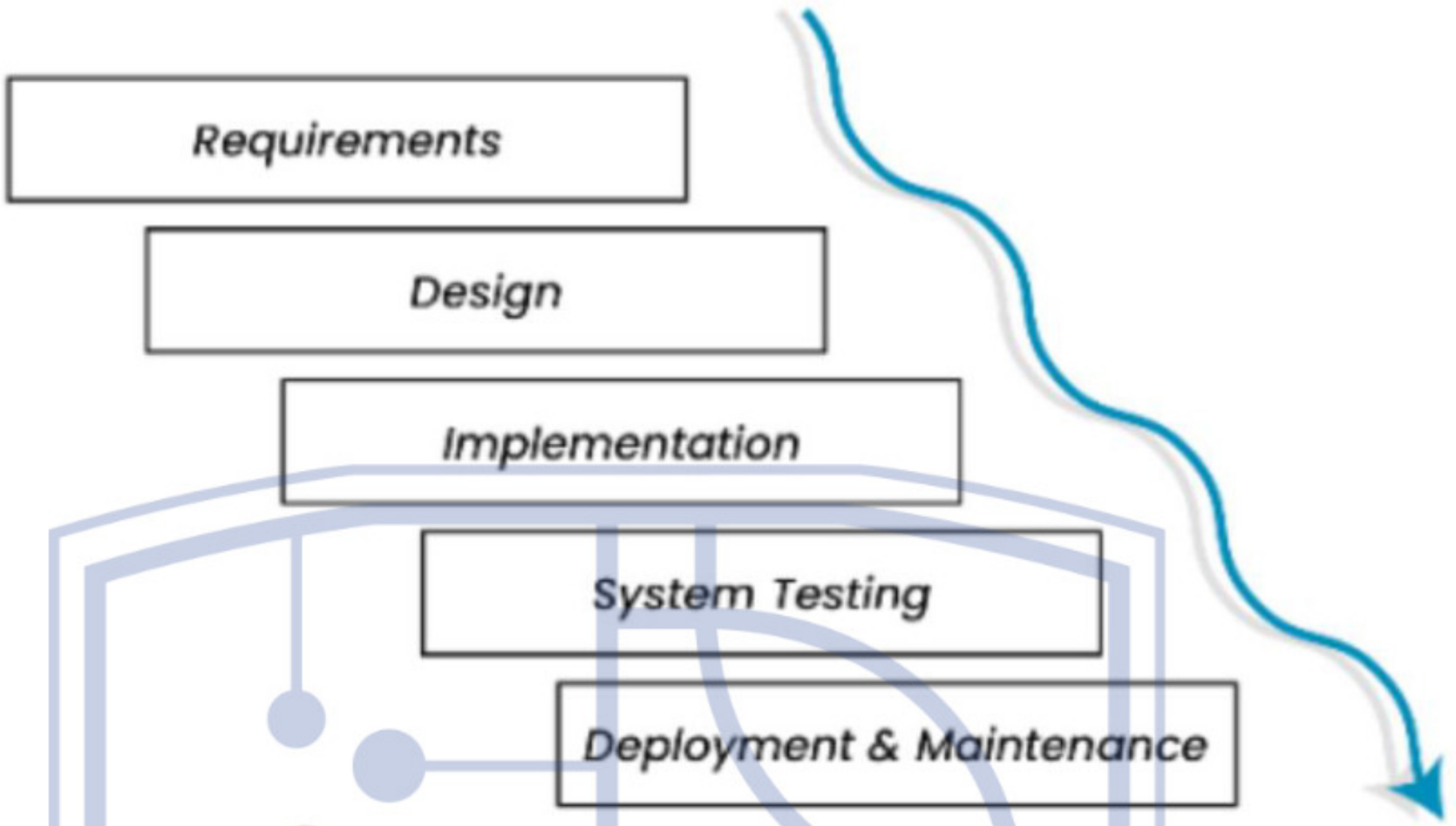
testing, *pairwise testing*, hingga pendekatan manual seperti *error guessing [38]*. Menghadapi banyaknya opsi metode tersebut, seorang penguji wajib untuk menyeleksi dan menentukan teknik pengujian mana yang paling relevan agar selaras dengan spesifikasi sistem yang diuji.

Sebagai salah satu teknik dalam *black box testing*, *Use Case Testing* merupakan pendekatan pengujian yang secara memanfaatkan skenario *use case* sebagai dasar untuk merancang dan mengeksekusi kasus uji [36]. Pendekatan ini memastikan bahwa perangkat lunak bekerja dengan baik sesuai dengan skenario penggunaan yang dirancang berdasarkan kebutuhan pengguna [39].Tujuan fundamental dari teknik ini adalah untuk memverifikasi bahwa sistem mampu memenuhi kebutuhan fungsional yang telah ditentukan dengan menguji interaksi antara aktor dan sistem [36]. Implementasi pengujian yang berpusat pada pengalaman interaksi nyata ini dinilai sangat krusial untuk memitigasi berbagai risiko kegagalan fungsional ketika perangkat lunak tersebut beroperasi di lingkungan publik [36].

Gambar 2. 4 Gambar *Black Box Testing* [40]

2.6 Waterfall

Model *waterfall* merupakan salah satu metode dalam *Software Development Life Cycle* (SDLC) yang menerapkan pendekatan sekuensial secara teratur untuk memproduksi perangkat lunak. Karakteristik utama dari metodologi ini menyerupai pergerakan air terjun, di mana setiap aktivitas pengerjaan dieksekusi secara linear berurutan mulai dari tahap perencanaan hingga tahap pemeliharaan operasional [41]. Alasan mendasar pemilihan model ini adalah kemampuannya dalam memfasilitasi manajemen proyek melalui tahapan yang terstruktur dan sistematis, terutama untuk sistem informasi dengan spesifikasi kebutuhan yang jelas sejak fase awal [42]. Pembagian fase yang kaku ini melarang adanya lompatan pengerjaan atau pengulangan kembali ke tahapan sebelumnya sebelum tahapan saat ini dinyatakan selesai. Dalam pengembangannya, tahapan-tahapan yang ada pada metode Waterfall antara lain [41]:



Gambar 2. 5 Gambar *Diagram* Alir Metode Waterfall *[41]*

1. *Requirement*

   Pada tahap ini, akan dilakukan analisa terhadap kebutuhan perangkat lunak yang akan dikembangkan, agar fitur dari sistem sesuai dengan yang diharapkan. Data diperoleh melalui studi literatur dan observasi terhadap sistem yang sudah ada, kemudian dianalisis untuk memperoleh informasi yang diperlukan.

2. *Design*

   Setelah kebutuhan sistem teridentifikasi, tahap ini berfokus pada penyusunan rancangan teknis perangkat lunak. Hal yang dirancang meliputi struktur arsitektur sistem, model basis data, serta tata letak antarmuka yang akan dilihat dan digunakan oleh pengguna. Rancangan ini menjadi acuan utama sebelum sistem masuk ke tahap pembuatan kode program.

3. *Implementation*

   Tahap ini merupakan inti dari keseluruhan proses pengembangan, yaitu proses sistem mulai dibangun menjadi sebuah perangkat lunak. Proses ini berupa penulisan kode program sesuai bahasa pemrograman yang dipilih. Keluaran dari tahap ini berupa aplikasi yang siap dijalankan dan digunakan sesuai dengan tujuan awal pengembangannya.

4. *System Testing*

Perangkat lunak yang baru selesai dibangun pada umumnya masih memiliki kekurangan, sehingga diperlukan tahap pengujian sebelum benar-benar siap digunakan. Pengujian ini bertujuan untuk memeriksa kesesuaian antara fitur yang berjalan dengan rancangan yang sudah ditentukan sebelumnya. Proses pengujian dapat dilakukan lebih dari satu kali hingga hasilnya dinilai memadai.

5. *Deployment & Maintenance*

Tahap terakhir ini dilakukan ketika aplikasi sudah selesai dibangun dan seluruh fungsinya telah berjalan dengan semestinya. Sistem kemudian dirilis agar bisa diakses oleh penggunanya. Selain itu, tahap ini juga mencakup kegiatan pemeliharaan secara berkelanjutan untuk menangani masalah yang mungkin muncul setelah sistem digunakan.

2.7 Lelang

Lelang merupakan sebuah teknik manajemen pendapatan yang terstruktur dan digunakan secara luas sebagai metode penjualan yang populer. Secara umum, lelang didefinisikan sebagai suatu mekanisme perdagangan yang terdiri dari seperangkat aturan formal untuk menjual produk kepada penawar dengan penawaran terbaik. Di dalam proses ini, setiap calon pembeli yang disebut sebagai penawar (*bidder*) akan menyampaikan pernyataan resmi mengenai harga yang rela mereka bayarkan, yang dikenal dengan istilah penawaran (*bid*). Barang yang dilelang kemudian biasanya akan dialokasikan kepada orang yang menawarkan harga tertinggi sebagai pemenang dari lelang tersebut [43].

Lelang berfungsi sebagai sarana penentu harga yang berjalan secara dinamis. Mekanisme ini berbeda dengan sistem penjualan biasa yang menerapkan harga pas, karena lelang membiarkan nilai suatu barang bergerak naik atau turun mengikuti kondisi pasar terkini. Mekanisme ini dinilai memiliki peran penting karena efisiensinya dalam mencocokkan ketersediaan pasokan dengan tingkat permintaan di pasar. Melalui rancangan mekanisme tersebut, alokasi barang serta penetapan harga akhirnya akan ditentukan sepenuhnya berdasarkan preferensi dan kesediaan membayar dari masing-masing peserta [43].

Walaupun lelang termasuk metode dagang yang sudah ada sejak lama, penerapannya berubah total sejak munculnya internet. Kehadiran *platform* digital berhasil mengubah cara kerja lelang tradisional yang dulunya mengharuskan semua orang berkumpul di satu lokasi fisik yang sama. Penggunaan jaringan internet ini membuat kegiatan lelang menjadi jauh lebih mudah diakses oleh siapa saja dan kapan saja tanpa sekat geografis [44]. Perubahan

dari sistem tatap muka menjadi sistem elektronik inilah yang memicu lahirnya lelang online sebagai bentuk modern dari mekanisme lelang konvensional.

2.7.1 Jenis – Jenis Lelang

Dalam sistem lelang, jenis mekanisme lelang menjadi dasar karena setiap jenis memiliki karakteristik yang berbeda. Pemilihan jenis lelang yang tepat akan mempengaruhi perilaku peserta, harga akhir, hingga keberhasilan lelang itu sendiri. Secara umum, terdapat tiga jenis lelang yang paling sering digunakan dalam sistem lelang online, yaitu *English Auction, Dutch Auction,* dan *Sealed-Bid Auction.*

1. *English Auction*

   English Auction merupakan metode lelang di mana penjual terlebih dahulu mendaftarkan barang yang akan dijual, lalu seluruh peserta lelang dapat melihat dan mengetahui setiap penawaran yang diajukan secara terbuka. Proses dimulai dengan penentuan harga awal atau harga minimum oleh penjual. Selanjutnya, para penawar akan saling menaikkan harga hingga tidak ada lagi peserta yang bersedia memberikan tawaran lebih tinggi. Pada titik tersebut, penawaran tertinggi menjadi harga akhir dan pemenang lelang [45], [46].

2. *Dutch Auction*

   Dutch Auction adalah metode lelang dengan harga yang dimulai dari nilai tinggi, kemudian diturunkan secara bertahap hingga ada peserta yang langsung menerima harga tersebut. Penawar pertama yang menyetujui harga saat itu akan memenangkan barang. Dalam mekanisme ini, peserta harus menentukan waktu yang tepat untuk menawar lebih cepat meningkatkan peluang menang tetapi berisiko membayar lebih mahal, sedangkan menunggu lebih lama bisa mendapatkan harga lebih rendah namun berisiko kalah dari peserta lain [46], [47].

3. *Sealed-Bid Auction*

   Sealed-bid auction adalah mekanisme lelang di mana setiap peserta mengajukan penawaran secara tertutup tanpa mengetahui nilai penawaran peserta lain. Seluruh bid dikumpulkan terlebih dahulu dan baru dibuka pada akhir periode lelang, sehingga proses berlangsung secara privat dan tidak ada interaksi langsung antar penawar selama berlangsungnya lelang.

   Salah satu bentuknya adalah *First-Price Sealed-Bid Auction (FPSB),* di mana pemenang ditentukan berdasarkan penawaran tertinggi dan harus membayar sesuai dengan nilai yang ia ajukan sendiri. Dalam mekanisme ini, peserta biasanya tidak

menawar tepat sesuai valuasi aslinya, melainkan sedikit lebih rendah untuk memperoleh keuntungan harga dengan tetap mempertimbangkan risiko kalah [46]. Sementara itu, *Second-Price Sealed-Bid Auction* atau *Vickrey auction* merupakan variasi di mana pemenang tetap adalah penawar tertinggi, namun harga yang dibayarkan adalah sebesar penawaran tertinggi kedua. Mekanisme ini mendorong peserta untuk menawar sesuai nilai sebenarnya karena tidak perlu khawatir membayar lebih tinggi dari yang diperlukan [46].

2.7.2 Mekanisme Lelang

Mekanisme lelang merujuk pada alur yang mengatur bagaimana suatu proses lelang berlangsung dari awal hingga selesai. Pada lelang online, mekanisme tersebut diimplementasikan dalam bentuk fitur dan alur sistem pada platform digital. Secara umum, mekanisme lelang online terdiri dari beberapa tahapan utama yang dijelaskan sebagai berikut.

1. Tahap Registrasi Pengguna

   Tahap registrasi merupakan tahap awal di mana calon pengguna mendaftarkan diri pada platform lelang untuk memperoleh hak berpartisipasi dalam proses lelang. Pada tahap ini, calon pengguna umumnya diminta untuk menyediakan informasi identitas yang akan diverifikasi oleh pihak platform. Proses verifikasi identitas bertujuan untuk memastikan bahwa setiap pengguna yang berpartisipasi dalam lelang merupakan pihak yang sah, sekaligus mencegah praktik-praktik kecurangan dan penyalahgunaan akun. Platform lelang online komersial seperti *eBay* dan *Catawiki* menerapkan proses verifikasi identitas dengan meminta pengguna menyediakan dokumen identitas resmi sebelum dapat melakukan transaksi pada platform [48], [49].

2. Tahap Pengajuan Objek Lelang

   Setelah pengguna terdaftar sebagai penjual, langkah berikutnya adalah pengajuan objek yang akan dilelang. Pada tahap ini, penjual menyediakan informasi mengenai objek lelang yang mencakup deskripsi, foto, kondisi, harga awal, serta informasi pendukung lainnya. Pada sebagian platform lelang online, objek yang diajukan oleh penjual harus melalui proses peninjauan dan persetujuan oleh pihak platform terlebih dahulu sebelum dapat dipublikasikan untuk dilelang. Sebagai contoh, Catawiki menerapkan proses peninjauan objek oleh tim ahli yang akan memutuskan apakah objek tersebut layak untuk dipublikasikan pada lelang [49].

3. Tahap Penawaran (*Bidding*)

Tahap penawaran merupakan tahap inti dari proses lelang di mana peserta saling mengajukan penawaran untuk memperoleh objek lelang. Pada lelang online, peserta mengajukan penawaran melalui antarmuka *platform*, kemudian penawaran tersebut dikirimkan ke *server* untuk diproses dan dicatat. Setiap penawaran yang masuk harus memenuhi aturan yang telah ditetapkan, seperti nilai penawaran yang lebih tinggi dari penawaran sebelumnya pada *English Auction*. Pergerakan harga pada tahap ini bergantung pada jenis lelang yang digunakan, misalnya naik secara bertahap pada *English Auction* atau turun secara bertahap pada *Dutch Auction*.

4. Tahap Penutupan dan Penentuan Pemenang

   Tahap penutupan terjadi ketika sesi lelang berakhir berdasarkan kondisi yang telah ditetapkan, baik karena durasi lelang telah habis maupun karena syarat tertentu telah terpenuhi. Setelah lelang ditutup, platform menentukan pemenang berdasarkan aturan yang berlaku, yang pada umumnya adalah peserta dengan penawaran tertinggi. Hasil lelang kemudian dipublikasikan kepada penjual dan pemenang sebagai dasar untuk melanjutkan ke tahap berikutnya.

5. Tahap Pasca-Lelang

   Tahap pasca-lelang mencakup proses pembayaran dari pemenang, pengiriman objek lelang dari penjual ke pemenang, serta penerusan dana dari platform kepada penjual. Pada platform lelang online, dana pembayaran dari pemenang umumnya ditahan terlebih dahulu oleh platform dan baru diteruskan kepada penjual setelah objek lelang diterima oleh pemenang. Mekanisme penahanan dana ini bertujuan untuk melindungi pemenang dari risiko objek lelang tidak dikirim atau tidak sesuai dengan deskripsi yang dijanjikan, seperti yang diterapkan pada *platform Catawiki* di mana pembayaran kepada penjual baru dilepaskan beberapa hari setelah objek diterima oleh pemenang [50].

2.8 Sistem Lelang *Online*

2.8.1 Pengertian Sistem Lelang *Online*

Sistem lelang *online* merupakan pengembangan dari sistem lelang tradisional, sistem lelang tradisional biasanya mengharuskan penjual dan pembeli untuk hadir di tempat yang sudah ditentukan dan dilakukan secara tatap muka, Sedangkan sistem lelang *online* bisa dilakukan dimana saja dan kapanpun melalui platform digital, sehingga penjual dan pembeli dapat berinteraksi tanpa batasan lokasi geografis [51], [52]. Dengan adanya sistem lelang

online, masalah seperti keterbatasan waktu dan tempat dapat diatasi sehingga memungkinkan lebih banyak peserta untuk ikut berpatisipasi.

Beberapa fungsionalitas juga diterapkan didalam sistem lelang *online* seperti mulai dari pendaftaran, proses penawaran, notifikasi dan pemberitahuan secara *real-time*, hingga penentuan pemenang [53]. Fitur yang penting dalam sistem lelang *online* adalah *Auction Management*. *Auction Management* berfungsi sebagai inti dari sistem lelang, di mana sistem mengatur keseluruhan proses kompetisi penawaran dalam periode waktu tertentu. Melalui *Auction Management*, sistem secara aktif melacak setiap tawaran yang masuk, memberi notifikasi kepada peserta lelang ketika ada tawaran baru yang masuk, dan secara otomatis menentukan pemenang dengan tawaran tertinggi saat lelang berakhir. Dengan demikian, setiap peserta dapat melakukan bid sambil memantau perkembangan harga tertinggi yang berlaku [51].

Secara umum, sistem lelang *online* dapat dipahami sebagai suatu platform digital yang dirancang untuk mengotomatisasi seluruh proses pelelangan agar berjalan secara efisien, transparan, dan *real-time*, sekaligus mengatasi keterbatasan waktu dan tempat yang terdapat pada sistem lelang tradisional.

2.8.2 Tren dan Perkembangan Sistem Lelang *Online* di Indonesia

Perkembangan sistem lelang *online* tidak terlepas dari pesatnya pertumbuhan teknologi internet dan transaksi digital. Di tingkat global, platform lelang online seperti *eBay* dan *Amazon* telah hadir sebagai sarana transaksi yang memfasilitasi mekanisme penawaran antara penjual dan pembeli secara daring. Di Indonesia, transformasi sistem lelang tradisional menjadi sistem lelang *online* diimplementasikan oleh Direktorat Jenderal Kekayaan Negara (DJKN) Kementerian Keuangan melalui peluncuran Portal Lelang Indonesia dengan domain lelang.go.id pada tahun 2018 [54]. Sejak peluncurannya, lelang.go.id menjadi platform digital yang digunakan untuk pelaksanaan berbagai jenis lelang yang dikelola oleh DJKN, baik lelang sukarela maupun lelang eksekusi. Sebagai gambaran skala aktivitas penyelenggaraan lelang yang dikelola oleh DJKN, capaian nilai transaksi lelang selama periode 2020–2023 dapat dilihat pada Gambar 2.5.

Gambar 2. 6 Statistik Nilai Transaksi Lelang Tahun 2020 – 2024 [55], [56], [57], [58], [59]

Pada Gambar 2.6, nilai transaksi lelang menunjukkan tren yang cenderung meningkat selama periode 2020–2023. Pada tahun 2020, nilai transaksi lelang tercatat sebesar 26,19 triliun rupiah, kemudian mengalami peningkatan signifikan pada tahun 2021 menjadi 35,16 triliun rupiah. Meskipun terjadi sedikit penurunan pada tahun 2022 menjadi 35 triliun rupiah, nilai transaksi kembali meningkat secara tajam pada tahun 2023 hingga mencapai 44,34 triliun rupiah. Sementara itu, pada tahun 2024 DJKN menetapkan target nilai transaksi sebesar 35 triliun rupiah, yang menunjukkan adanya upaya pengendalian dan perencanaan pertumbuhan transaksi lelang secara lebih terukur. Sebagai bagian dari upaya peningkatan kualitas layanan lelang online, DJKN melakukan pengembangan platform lelang.go.id dengan menghadirkan lelang.go.id Versi 2 yang diimplementasikan secara bertahap sejak pertengahan tahun 2024 [60]. Versi terbaru ini menghadirkan antarmuka yang lebih modern, fitur pencarian yang lebih canggih, serta integrasi data yang lebih luas dalam pelaksanaan lelang secara daring. Pembaruan platform ini menunjukkan adanya komitmen pemerintah untuk terus mengembangkan sistem lelang online sebagai sarana transaksi digital yang adaptif terhadap kebutuhan masyarakat.

2.8.3 Karakteristik dan Komponen Sistem Lelang *Online*

Sistem lelang *online* merupakan hasil dari kombinasi antara teknologi dan sistem lelang tradisional yang bertujuan untuk menciptakan proses transaksi yang lebih cepat, efisien, dan transparan. Sistem lelang *online* memiliki sejumlah karakteristik utama yang menjadi pembeda dari model sistem lelang tradisional, seperti aksesibilitas global,

otomatisasi proses penawaran, transparansi digital, serta dukungan teknologi untuk mendukung kelancaran transaksi.

Karakteristik Sistem Lelang *Online* mencakup beberapa hal penting yaitu:

1. Aksesibilitas Global

   Sistem lelang *online* memberikan kemudahan bagi pengguna, karena pengguna dapat mengikuti proses pelelangan tanpa harus hadir secara fisik di lokasi tertentu. Melalui koneksi internet, aktivitas lelang dapat dilakukan kapan saja dan di mana saja [51], [52], sehingga mengatasi keterbatasan waktu dan tempat yang biasanya terjadi pada sistem lelang tradisional. Melalui sistem ini, jumlah pengguna yang terlibat semakin luas, interaksi antarwilayah menjadi lebih mudah, serta meningkatkan potensi transaksi yang terjadi dalam satu periode lelang.

2. Proses Otomatis dan Waktu-Nyata (*Real-Time*)

   Sistem lelang *online* memiliki keunggulan lain yaitu kemampuannya menjalankan proses secara otomatis dan waktu-nyata (*real-time*). Setiap penawaran yang dilakukan oleh pengguna akan langsung tercatat secara otomatis oleh sistem dan akan langsung ditampilkan secara instan kepada peserta lain yang mengikuti lelang. Fitur lainnya seperti notifikasi berguna untuk memberitahu peserta lelang mengenai tawaran baru, hingga penutupan lelang juga dilakukan otomatis oleh sistem [53]. Hal ini dapat membuat pengguna menjadi lebih interaktif dan responsif terhadap perubahan harga secara langsung. Dengan adanya sistem seperti ini, proses lelang menjadi lebih aktif dan menarik karena seluruh peserta mendapatkan informasi yang sama secara *real time*.

3. Integrasi Teknologi Pendukung

   Sistem lelang *online* menerapkan berbagai teknologi pendukung seperti notifikasi waktu nyata serta sistem pembayaran digital yang diimplementasikan untuk menciptakan pengalaman transaksi yang lebih menyeluruh. Dengan penerapan berbagai teknologi tersebut, platform lelang online tidak hanya berfungsi sebagai tempat penawaran harga, tetapi juga sebagai sistem perdagangan digital yang lengkap mulai dari pendaftaran peserta hingga proses pembayaran [53].

2.8.4 Permasalahan dalam Sistem Lelang *Online*

Sistem lelang *online* meskipun menawarkan berbagai keunggulan seperti efisiensi, keterbukaan, dan kemudahan akses, tetap memiliki sejumlah tantangan teknis yang perlu diperhatikan untuk menjamin keandalan serta keadilan dalam prosesnya. Salah satu permasalahan yang cukup krusial adalah terjadinya *race condition* pada mekanisme *real-time bidding*. Kondisi ini muncul ketika banyak peserta mengajukan tawaran secara hampir bersamaan, terutama menjelang akhir periode lelang. Sistem harus mampu memproses setiap tawaran agar tidak terjadi duplikasi pemenang.

Situasi tersebut biasanya disebabkan oleh adanya pembaruan nilai tawaran tertinggi (*current highest bid*) yang dilakukan secara bersamaan oleh banyak transaksi tanpa adanya pengendalian yang memadai. Apabila sistem tidak menggunakan mekanisme seperti *pessimistic locking*, maka dua atau lebih proses dapat bersaing untuk memperbarui data yang sama pada waktu yang hampir bersamaan. Hal ini dapat menyebabkan ketidakkonsistenan hasil dalam proses lelang. Karena itu, rancangan arsitektur sistem lelang *online* harus mampu menangani transaksi yang berjalan secara paralel (*multiple concurrent transactions*) [5] .

Selain itu, dalam sistem lelang *online* modern, urutan kedatangan tawaran memiliki peran penting dalam menentukan pemenang, terutama ketika dua penawaran bernilai sama diterima hampir bersamaan. Ketepatan sistem dalam mencatat waktu dan memproses urutan penawaran menjadi faktor utama agar hasil lelang tetap transparan.

Secara keseluruhan, *race condition* dapat menimbulkan dampak negatif berupa hasil lelang yang tidak akurat, potensi ketidakadilan di antara peserta. Oleh karena itu, penerapan mekanisme seperti *pessimistic locking*, pengelolaan transaksi yang terstruktur, dan pencatatan aktivitas (*audit trail*) yang transparan menjadi aspek penting dalam menjaga integritas dan keandalan sistem lelang *online*.

2.9 Website

Website merupakan kumpulan halaman digital yang berisi informasi dalam berbagai bentuk, seperti teks, gambar, animasi, suara, video, atau kombinasinya. Informasi tersebut disediakan melalui jalur koneksi internet sehingga dapat diakses oleh pengguna di seluruh dunia [61], [62]. Selanjutnya, *website* mengadopsi konsep *hyperlink* yang memungkinkan pengguna menelusuri informasi dari satu halaman ke halaman lain dengan mudah [62]. Selain itu, *website* juga dipahami sebagai kumpulan halaman yang terangkum dalam suatu domain atau subdomain di dalam *World Wide Web* (WWW) [63].

Secara umum, *website* dibagi menjadi dua jenis utama, yaitu *website* statis dan *website* dinamis. *Website* statis merupakan *website* yang menampilkan konten yang tetap dan jarang mengalami perubahan, sehingga informasi yang ditampilkan kepada setiap pengguna cenderung sama [63]. Sementara itu, *website* dinamis adalah *website* yang kontennya dapat berubah-ubah dan disesuaikan dengan interaksi atau preferensi pengguna, sehingga lebih interaktif dan fleksibel dalam penyajian informasi [63].

Selain berdasarkan sifatnya, website juga dapat dikelompokkan berdasarkan kegunaannya [63]:

1. *Website* pribadi, yaitu website yang berisi informasi pribadi dan dikelola oleh individu, seperti portofolio atau personal branding.
2. *Website* e-commerce, yaitu website yang digunakan sebagai sarana transaksi jual beli secara online.
3. *Website* bisnis, yaitu website yang digunakan sebagai media pemasaran dan promosi suatu perusahaan atau organisasi.
4. *Website* media sosial, yaitu website yang digunakan untuk berkomunikasi dan berinteraksi antar pengguna secara digital.
5. *Blog*, yaitu *website* yang berfungsi sebagai jurnal online untuk menulis artikel, berbagi informasi, atau ulasan.

Website juga memiliki berbagai manfaat, di antaranya sebagai media untuk membangun personal branding, meningkatkan kepercayaan pengguna atau pelanggan, sarana promosi, sarana menjual barang secara *online*, media berbagi informasi, serta sebagai platform untuk kegiatan sosial seperti penggalangan dana [63]. Berdasarkan berbagai manfaat tersebut, *website* juga menjadi komponen utama dalam mendukung implementasi sistem lelang *online* berbasis *web*. *Website* tidak hanya berfungsi sebagai media informasi, tetapi juga sebagai sarana interaksi antara penjual dan peserta lelang dalam melakukan proses penawaran secara transparan dan terstruktur. Melalui website, pengguna dapat mengakses informasi barang lelang, mengikuti proses *bidding* secara *real-time*, serta memantau hasil lelang dengan mudah. Dengan dukungan website dinamis, sistem lelang *online* mampu menyajikan data yang selalu diperbarui sesuai aktivitas pengguna, sehingga meningkatkan efisiensi, dan kemudahan dalam pelaksanaan lelang secara digital.

2.10 Database

Database merupakan sekumpulan informasi yang tersimpan secara sistematis di dalam komputer, sehingga dapat diakses dan diolah melalui program komputer untuk

memperoleh informasi yang dibutuhkan [64]. Konsep dasar dari *database* adalah kumpulan informasi atau entitas data yang saling terkait. Setiap *database* memiliki penjelasan terstruktur mengenai jenis data yang disimpan di dalamnya, yang disebut sebagai skema. Skema berfungsi untuk menggambarkan data-data yang diwakili oleh suatu database beserta hubungan antar setiap data tersebut [64].

Perangkat lunak yang bertugas mengelola serta memanggil kueri atau *query* dari sebuah *database* dikenal sebagai sistem manajemen basis data atau *Database Management System* (DBMS). Perangkat lunak ini bertanggung jawab untuk mengatur basis data secara efektif dan efisien, mulai dari tahap pembuatan awal hingga pengelolaan operasionalnya . Fungsi operasional tersebut mencakup proses memasukkan, mengubah, menghapus data, hingga menghasilkan laporan operasional terkait informasi yang dibutuhkan [65]. Secara lebih luas, DBMS adalah kombinasi antara perangkat keras dan perangkat lunak komputer yang khusus didesain untuk memberikan perlindungan serta mengelola database [64].

Dalam menjaga dan memelihara integritas data di dalam suatu sistem, terdapat beberapa fungsi utama yang dimiliki oleh DBMS, antara lain [64]:

1. Menjaga Integritas Data: DBMS bertugas mengurangi serta menghilangkan redundansi atau duplikasi data, sekaligus memaksimalkan konsistensi data agar setiap data yang ditampilkan selalu sesuai dengan data aslinya.
2. Penyimpanan Data (*Data Storage Management*): Fungsi utama DBMS adalah sebagai tempat penyimpanan data, termasuk berbagai jenis data seperti video dan gambar.
3. Keamanan Data: DBMS memiliki peran penting dalam mengatur pemberian hak akses hanya kepada orang-orang yang berwenang, serta menentukan tindakan apa saja yang dapat dilakukan oleh pengguna terhadap sebuah database.
4. Menyediakan Prosedur *Backup* dan *Recovery*: DBMS menyediakan fasilitas untuk melakukan pencadangan (*backup*) dan pemulihan (*recovery*) database sesuai kebutuhan dengan memanfaatkan teknik yang dimiliki oleh masing-masing DBMS.
5. Manajemen Transaksi: DBMS menyediakan mekanisme untuk mengatur berbagai transaksi dan perintah yang disampaikan, guna memastikan konsistensi data tetap terjaga.