

SKRIPSI

STUDI DAN ANALISIS KINERJA INDEXEDDB PADA CLIENT
WEB BROWSER

Disusun dan diajukan oleh:

WINDA ASTIYANTI AZIS
D421 14 015



DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
MAKASSAR
2021

LEMBAR PENGESAHAN SKRIPSI

STUDI DAN ANALISIS KINERJA INDEXEDDB PADA CLIENT WEB BROWSER

Disusun dan diajukan oleh:

WINDA ASTIYANTI AZIS

D421 14 015

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi

Program Sarjana Program Studi Teknik Informatika

Fakultas Teknik Universitas Hasanuddin

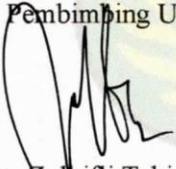
Pada tanggal 6 Agustus 2021

dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,

Pembimbing Pendamping,


Dr. Eng. Zulkifli Tahir, S.T., M.Sc.
NIP. 19840403 201012 1 004


A. Ais Prayogi Alimuddin, S.T., M.Eng.
NIP. 19830510 201404 1 001



Ketua Program Studi


Dr. Amil Annyad Ilham, S.T., M.IT
NIP. 19731010 199802 1 001

PERNYATAAN KEASLIAN

Yang bertanda tangan dibawah ini:

Nama : WINDA ASTIYANTI AZIS

Nim : D421 14 015

Program Studi : S1 Teknik Informatika

Menyatakan dengan sebenar-benarnya bahwa skripsi yang berjudul:

STUDI DAN ANALISIS KINERJA INDEXEDDB PADA CLIENT WEB BROWSER

Adalah karya ilmiah saya sendiri dan sepanjang pengetahuan saya di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan/ditulis/diterbitkan sebelumnya, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila dikemudian hari ternyata didalam naskah skripsi ini terdapat unsur-unsur djiplakan, saya bersedia menerima sanksi atas perbuatan tersebut dan diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2000, pasal 25 ayat 2 dan pasal 70).

Makassar, 12 Agustus 2021

Yang membuat Pernyataan



WINDA ASTIYANTI AZIS

ABSTRAK

Dalam pengembangan web, terdapat istilah *client side* dan *server side*. Hal ini merujuk pada sebuah proses yang dilakukan pada sisi *client* atau di sisi *server*. Untuk *client side*, semua proses terjadi di sisi pengguna, *client* meminta data ke *server* dimana data yang dikirimkan nanti diolah di sisi *client*. Biasanya data yang diolah dalam bentuk *HTML*, *CSS*, dan *JavaScript*. Beberapa tahun terakhir, sejumlah mekanisme canggih untuk menyimpan dan mengelola data pada web *client* sangat beragam salah satunya adalah *IndexedDB*. *IndexedDB* merupakan sistem penyimpanan lokal berbasis *NoSQL* di *browser* yang didukung oleh *HTML5* yang dapat menyimpan data apapun di *browser* pengguna untuk keperluan aplikasi. Dalam penelitian ini, analisis dilakukan terhadap kinerja *IndexedDB* sebagai *client side database* di beberapa *browser* yang populer digunakan, kemudian eksperimen dilakukan dalam beberapa skenario pengujian. Indikator yang diperhatikan dalam penelitian ini adalah *response time* dan proses *browser*. Hasil penelitian menunjukkan bahwa kinerja *IndexedDB* di sisi *client* bisa menunjukkan hasil yang berbeda pada setiap *browser* yang digunakan namun hal ini tidak dipengaruhi oleh perbedaan spesifikasi perangkat keras yang digunakan.

Kata kunci: *client side, database, HTML5, IndexedDB, browser.*

KATA PENGANTAR

Bismillahirrahmanirrahim.

Assalamu 'alaikum Warahmatullahi Wabarakatuh.

Segala puji dan syukur atas kehadiran Allah SWT yang telah melimpahkan Rahmat dan Karunia-Nya sehingga Tugas Akhir dengan judul **“Studi dan Analisis Kinerja IndexedDB pada Client Web Browser”** ini dapat diselesaikan sebagai salah satu syarat dalam menyelesaikan jenjang Strata-1 pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Pada penyusunan kali ini disajikan hasil penelitian menyangkut judul yang telah diangkat dan telah melalui proses pencarian dari berbagai sumber baik jurnal penelitian, prosiding pada seminar-seminar nasional/internasional, buku maupun dari situs-situs di internet, selain dari hasil-hasil penelitian sebelumnya.

Penulis menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai dengan masa penyusunan Tugas Akhir, sangatlah sulit untuk menyelesaikan Tugas Akhir ini. Oleh karena itu, penulis berterima kasih kepada:

1. Allah SWT, atas segala rahmat dan karunia-Nya yang telah diberikan kepada penulis.
2. Kedua orang tua, saudari, dan seluruh keluarga, atas segala doa, dukungan material, serta motivasi yang telah diberikan selama ini.
3. Bapak Dr. Eng. Zulkifli Tahir, S.T., M.Sc. dan Bapak A. Ais Prayogi Alimuddin, S.T., M.Eng., selaku dosen pembimbing.

4. Bapak Dr. Amil Ahmad Ilham, S.T., M.IT., selaku dosen penguji dan Ketua Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.
5. Bapak Dr. Eng. Muhammad Niswar, S.T., M.IT., selaku dosen penguji dan kepala Laboratorium Ubiquitous Computing & Networking (UbiCON).
6. Bapak Dr. Indrabayu, S.T., M.T., M.Bus.Sys., selaku dosen penguji.
7. Bapak dan Ibu dosen serta seluruh staf Departemen Teknik Informatika Universitas Hasanuddin yang telah membantu penulis.
8. Keluarga Angkatan 2014 Departemen Teknik Informatika FT UH atas segala bantuan dan semangat yang diberikan selama ini.
9. Teman-teman Rectifier FT UH atas dukungan dan semangat yang diberikan selama ini.
10. Para sahabat-sahabat dekat penulis dan kepada seluruh pihak yang berpengaruh lainnya yang tidak dapat saya sebutkan satu per satu yang tanpa sadar telah banyak memberi semangat dan inspirasi kepada penulis.

Akhir kata, penulis berharap semoga Allah SWT senantiasa membalas segala kebaikan dari semua pihak yang telah banyak membantu. Semoga tugas akhir ini dapat memberikan manfaat bagi pengembangan ilmu selanjutnya. Aamiin. *Wassalamu'alaikum Warahmatullahi Wabarakatuh.*

Makassar, Juli 2021

Penulis

DAFTAR ISI

HALAMAN JUDUL.....	i
LEMBAR PENGESAHAN	ii
PERNYATAAN KEASLIAN.....	iii
ABSTRAK	iv
KATA PENGANTAR	v
DAFTAR ISI.....	vii
DAFTAR GAMBAR	x
DAFTAR TABEL	xii
BAB I PENDAHULUAN	1
1.1. Latar Belakang.....	1
1.2. Rumusan Masalah.....	2
1.3. Tujuan Penelitian	3
1.4. Manfaat Penelitian	3
1.5. Batasan Masalah	3
1.6. Sistematika Penulisan	4
BAB II TINJAUAN PUSTAKA.....	6
2.1. Aplikasi Web	6
2.2. <i>Client Side Storage</i>	9
2.3. <i>Browser</i>	10
2.3.1 <i>Google Chrome</i>	11
2.3.2 <i>Mozilla Firefox</i>	12

2.3.3	<i>Opera Browser</i>	14
2.3.4	<i>Microsoft Edge</i>	15
2.4.	HTML	17
2.5.	<i>Database</i>	18
2.5.1	<i>SQL dan NoSQL database</i>	19
2.5.2	<i>Server-side dan client-side database</i>	20
2.5.3	<i>Tinjauan umum tentang database sisi client</i>	21
2.6.	DexieJS	23
2.7.	<i>IndexedDB</i>	24
BAB III METODOLOGI PENELITIAN.....		32
3.1.	Waktu dan Lokasi Penelitian	32
3.2.	Instrumen Penelitian	32
3.3.	Tahapan Penelitian.....	33
3.4.	Tahapan Persiapan	35
3.5.	Gambaran Umum.....	35
3.6.	Hasil Pembuatan Sistem	40
3.7.	Skenario Pengujian	41
3.7.1.	<i>Respon Time</i>	41
3.7.2.	<i>Proses Browser</i>	42
BAB IV HASIL DAN PEMBAHASAN		44
4.1.1	Hasil Pengujian Pada Perangkat 1	44
4.1.2	Hasil Pengujian Pada Perangkat 2	46
4.2.1	Hasil Pengujian Pada Perangkat 1	48

4.2.2	Hasil Pengujian Pada Perangkat 2	51
BAB V PENUTUP.....		54
5.1	Kesimpulan	54
5.2	Saran	55
DAFTAR PUSTAKA		56
LAMPIRAN.....		59

DAFTAR GAMBAR

Gambar 2. 1 Alur kerja aplikasi web	6
Gambar 2. 2 Model interaksi konvensional	7
Gambar 2. 3 Model interaksi modern	8
Gambar 2. 4 Browser google chrome	11
Gambar 2. 5 Browser mozilla firefox	13
Gambar 2. 6 Browser opera	14
Gambar 2. 7 Browser microsoft edge	16
Gambar 2. 8 Dokumen html sederhana.....	17
Gambar 2. 9 Grafik survei penggunaan database	19
Gambar 2. 10 Implementasi IndexedDB.....	25
Gambar 2. 11 Sampel data	25
Gambar 2. 12 Open DB, create, tambah tabel	26
Gambar 2. 13 Browser support	27
Gambar 3. 1 Alur tahapan penelitian	33
Gambar 3. 2 Gambaran umum kerja sistem.....	36
Gambar 3. 3 Diagram halaman utama	37
Gambar 3. 4 Diagram fitur create	37
Gambar 3. 5 Diagram fitur read.....	38
Gambar 3. 6 Diagram fitur update	39
Gambar 3. 7 Diagram fitur delete	39

Gambar 3. 8 Tampilan sistem	40
Gambar 3. 9 Basis data lokal IndexedDB	40
Gambar 3. 10 Dev tools google chrome	43
Gambar 4. 1 Grafik nilai rata-rata respon time dari basic operation.....	44
Gambar 4. 2 Grafik nilai rata-rata respon time dari crud operations	45
Gambar 4. 3 Grafik nilai rata-rata respon time dari basic operations	46
Gambar 4. 4 Grafik nilai rata-rata respon time dari crud operations	47
Gambar 4. 5 Grafik prosess browser untuk waktu DomContentLoaded.....	50
Gambar 4. 6 Grafik prosess browser untuk waktu load.....	50
Gambar 4. 7 Grafik prosess browser untuk waktu DomContentLoaded.....	52
Gambar 4. 8 Grafik prosess browser untuk waktu load.....	53

DAFTAR TABEL

Tabel 2. 1 Versi html	18
Tabel 3. 1 Format basis data lokal	41
Tabel 4. 1 Tabel hasil pengujian akses pertama kali tanpa cache.....	49
Tabel 4. 2 Tabel pengujian dengan cache	49
Tabel 4. 3 Tabel hasil pengujian akses pertama kali tanpa cache.....	51
Tabel 4. 4 Tabel pengujian dengan cache	52

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dalam pengembangan web, terdapat istilah *client side* dan *server side*. Hal ini merujuk pada sebuah proses yang dilakukan pada sisi *client* atau di sisi *server*. Untuk *client side*, semua proses terjadi di sisi pengguna, *client* meminta data ke *server* dimana data yang dikirimkan nanti diolah di sisi *client*. Biasanya data yang diolah dalam bentuk *HTML*, *CSS*, dan *JavaScript*.

Sedangkan untuk *server side*, semua proses yang terjadi dilakukan di sisi *server*. *Server side* bertanggung jawab untuk merespon data yang diminta oleh *client side*. Biasanya *server side* merupakan tempat dimana terjadinya interaksi pada *database*, sehingga sisi *client* tidak mengetahui prosesnya dan memang tidak boleh tahu. *Client* hanya diberikan sebuah data hasil olahan dari sisi *server*.

Database adalah bagian penting dari aplikasi web berbasis internet saat ini. Untuk saat ini, hampir semua aplikasi web menggunakan *database* sisi *server*. *Database* sisi *client* memiliki keuntungan dapat mengurangi beban pada *server* web, tetapi kinerja *database* akan bervariasi tergantung pada web *browser* pengguna dan khususnya bagaimana desainer *browser* memilih untuk menerapkan penyimpanan lokal.

Permasalahan yang sering muncul saat ini pada penyimpanan di sisi web *server* yaitu: memerlukan koneksi internet, terlalu membebani *server*, dan transfer data antara *server* dan *client* membutuhkan waktu yang cukup lama sehingga resiko

melambatnya situs web sering terjadi. Dengan menggunakan mekanisme penyimpanan sisi *client*, sebagian besar dari kerugian tersebut bisa diatasi, ini tidak hanya membuat situs web lebih cepat, tetapi juga memungkinkan dukungan *offline* dan mengurangi beban *server*.

Browser sendiri menyediakan beberapa cara bagi aplikasi Anda untuk menyimpan data di sisi *client*. Satu opsi penyimpanan adalah *IndexedDB*, standar web yang didukung oleh beberapa *browser*. Pada tahun 2009, *Oracle* menawarkan *IndexedDB* sebagai penyimpanan lokal baru pada web *browser* standar. *IndexedDB* menyediakan implementasi basis data terindeks, dimana setiap *record* diidentifikasi oleh indeks atau kunci unik, membuat pengambilan data menjadi cepat. Pada Januari 2017, *IndexedDB* terus memperbarui versinya dengan menambahkan beberapa fitur *update* dan dan sampai saat ini terus dilakukan pengembangan fungsi-fungsi dari versi sebelumnya. *IndexedDB* memperkenalkan beberapa fungsi yang mengoptimalkan beberapa pola akses basis data dan meningkatkan ergonomi pemrograman.

Berdasarkan hal tersebut, penulis tertarik melakukan sebuah penelitian dengan judul “**Studi dan Analisis Kinerja *IndexedDB* pada *Client Web Browser***”. Dengan harapan, penelitian ini dapat menjadi salah satu kerangka acuan dalam implementasi penggunaan penyimpanan lokal sesuai kebutuhan.

1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan di atas, maka pokok

permasalahan dalam penelitian ini yaitu:

1. Bagaimana implementasi *IndexedDB* pada web *client*?
2. Bagaimana hasil kinerja *IndexedDB* sebagai penyimpanan lokal pada sistem web *client* di masing-masing *browser* yang berbeda?

1.3. Tujuan Penelitian

Berdasarkan rumusan masalah yang telah diuraikan, maka tujuan dari penelitian ini yaitu:

1. Mengetahui implementasi *IndexedDB* pada web *client* sebagai penyimpanan data sisi *client*.
2. Mengetahui dan membandingkan kinerja *IndexedDB* sebagai penyimpanan lokal pada sistem web *client* di masing-masing *browser* yang berbeda.

1.4. Manfaat Penelitian

Manfaat dari penelitian ini adalah:

1. Dapat mengetahui implementasi *IndexedDB* pada web *client*.
2. Dapat mengetahui kinerja *IndexedDB* sebagai penyimpanan lokal pada sistem web *client* dan kita dapat melihat perbandingan performa dari setiap web *browser*.

1.5. Batasan Masalah

Yang menjadi batasan masalah dari penelitian ini adalah:

1. Analisis dilakukan pada sistem web *client* dengan melakukan implementasi pada penyimpanan data lokal menggunakan *IndexedDB*.
2. Web *browser* yang dipilih untuk analisis dalam studi ini yaitu *Google Chrome*, *Mozilla Firefox*, *Opera Browser*, dan *Microsoft Edge*.
3. Hal yang menjadi objek utama dalam analisis kinerja *IndexedDB* yaitu untuk menjalankan beberapa operasi *database* (*open database*, *create object store*, *close database*, *create record*, *read record*, *update record*, dan *delete record*).
4. Parameter dari analisis kinerja yang akan dilakukan yaitu *response time* dan proses *browser*.

1.6. Sistematika Penulisan

Untuk memberikan gambaran singkat mengenai isi tulisan secara keseluruhan, maka akan diuraikan beberapa tahapan dari penulisan secara sistematis, yaitu:

BAB I: PENDAHULUAN

Bab ini menguraikan secara singkat latar belakang penelitian, rumusan dan batasan masalah, tujuan dan manfaat penelitian, dan sistematika penulisan.

BAB II: TINJAUAN PUSTAKA

Bab ini membahas kerangka berpikir, serta landasan teori yang berhubungan dengan studi dan analisis yang dilakukan dalam penelitian ini.

BAB III: METODOLOGI PENELITIAN

Bab ini membahas tentang metode pengujian yang di terapkan dalam analisis ini.

BAB IV: HASIL DAN PEMBAHASAN

Bab ini berisi hasil penelitian dan pembahasan penjabaran dari penelitian yang dilakukan.

BAB V: PENUTUP

Bab ini berisi tentang kesimpulan yang didapatkan berdasarkan hasil penelitian yang telah dilakukan serta saran-saran untuk pengembangan lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1. Aplikasi Web

Aplikasi web merupakan aplikasi yang diakses menggunakan web *browser* melalui jaringan internet. Aplikasi web menggunakan kombinasi skrip sisi *server* (*PHP* dan *ASP*) untuk menangani penyimpanan dan pengambilan informasi, dan skrip sisi *client* (*JavaScript* dan *HTML*) untuk menyajikan informasi kepada pengguna (Camden, 2015).



Gambar 2. 1 Alur kerja aplikasi web

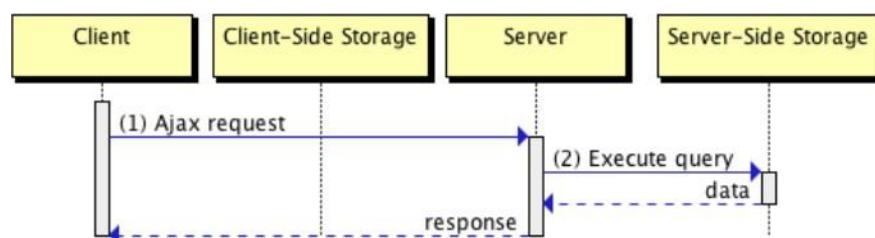
Adapun cara kerja web adalah sebagai berikut:

- Informasi web disimpan dalam bentuk halaman-halaman web.
- Halaman web tersebut disimpan dalam computer *server* web.
- Sementara dipihak pemakai ada computer yang bertindak sebagai komputer *client* dimana ditempatkan program untuk membaca halaman web yang ada di *server* web (*browser*).
- *Browser* membaca halaman web yang ada di *server* web.

Ada 2 bagian pokok dalam aplikasi web, yang pertama adalah sisi *client* dan yang kedua adalah sisi *server*, sisi *client* dalam hal ini adalah *PC* atau bisa juga

perangkat *mobile* yang terhubung ke jaringan internet, *client* dapat mengakses aplikasi web melalui web *browser* seperti *Mozilla Firefox*, *Google Chrome*, *Opera Browser* dan lain-lain. Sedangkan *server* adalah perangkat komputer dengan spesifikasi yang bagus digunakan untuk menyimpan aplikasi web beserta *database server* yang siap untuk diakses oleh *client*, *client* bertugas meminta halaman web *server* melalui web *browser*, web *browser* akan meneruskannya ke *server* dimana aplikasi web berada, komputer *server* akan mengolah permintaan dari *client*, ketika halaman web yang diminta ditemukan maka komputer *server* akan mengirimkannya ke komputer *client* dan halaman web yang diminta akan ditampilkan pada web *browser* di komputer *client* (Camden, 2015).

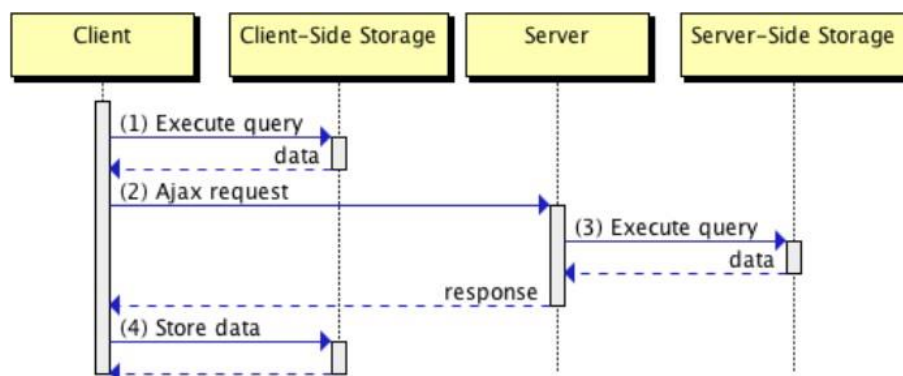
Aplikasi web memiliki dua opsi untuk menyimpan data: (1) di web *server* atau (2) di *client* web (pengguna akhir komputer). Tipe data tertentu termasuk dalam satu, sementara yang lain bekerja lebih baik di sisi lainnya. Sebagai contoh, data yang sensitif dan rapuh harus selalu disimpan di *server*, sedangkan data statis dan preferensi pengguna dapat disimpan pada *client* (Laine, 2012).



Gambar 2. 2 Model interaksi konvensional

Gambar 2.2 mengilustrasikan model interaksi konvensional, dimana semua data disimpan di penyimpanan sisi *server* (misalnya, *database*). Ketika seorang

pengguna meminta data baru, permintaan *Ajax* dibuat dan data diambil dari *database* yang berada di *server*. Kerugian dari pendekatan ini adalah: (1) lalu lintas jaringan yang tidak perlu, (2) membutuhkan koneksi internet, dan (3) menjadi beban *server load*. Mentransfer data antara *server* dan *client*, tentu saja, membutuhkan waktu, dan dengan demikian memperlambat situs web, yang mengarah ke pengalaman pengguna yang buruk (Laine, 2012).



Gambar 2. 3 Model interaksi modern

Dengan menggunakan mekanisme penyimpanan sisi *client*, sebagian besar kerugian yang disebutkan di atas dapat diatasi. Gambar 2.3 mengilustrasikan model interaksi modern, yang mencakup penyimpanan sisi *client*. Pertama, halaman web memeriksa apakah penyimpanan sisi *client* sudah memiliki data baru yang diminta. Jika ya, maka data segera ditampilkan pada antarmuka pengguna tanpa berkomunikasi dengan *server*. Jika tidak, data baru yang diminta diambil dari *database* yang berada di *server*, dan ketika diperoleh, data disimpan di penyimpanan sisi *client* untuk digunakan di masa mendatang. Pendekatan ini tidak hanya membuat situs web lebih cepat, tetapi juga memungkinkan dukungan *offline* dan mengurangi beban *server* (Laine, 2012).

Cookie HTTP adalah mekanisme pertama, yang memungkinkan aplikasi web untuk menyimpan data pada web *client*. Namun, *Cookie* tidak memenuhi persyaratan aplikasi web modern (Laine, 2012).

2.2. *Client Side Storage*

Sebagian besar situs web modern bersifat dinamis - mereka menyimpan data di *server* menggunakan beberapa jenis basis data (penyimpanan sisi *server*), kemudian menjalankan kode sisi *server* untuk mengambil data yang dibutuhkan, memasukkannya ke dalam *template* laman statis, dan menayangkan *HTML* yang dihasilkan ke *client* untuk ditampilkan oleh *browser* pengguna (Camden, 2015).

Penyimpanan sisi *client* berfungsi dengan prinsip yang sama, tetapi memiliki kegunaan yang berbeda. Ini terdiri dari JavaScript API yang memungkinkan Anda untuk menyimpan data pada *client* (yaitu pada mesin pengguna) dan kemudian mengambilnya saat diperlukan. Ini memiliki banyak kegunaan yang berbeda, seperti:

- Mempersonalisasi preferensi situs (mis. menunjukkan pilihan pengguna untuk widget khusus, skema warna, atau ukuran font).
- Menahan aktivitas situs sebelumnya (mis. menyimpan konten keranjang belanja dari sesi sebelumnya, mengingat jika sebelumnya ada pengguna yang masuk).
- Menyimpan data dan aset secara lokal sehingga situs akan lebih cepat (dan berpotensi lebih murah) untuk diunduh, atau dapat digunakan tanpa koneksi jaringan.
- Menyimpan aplikasi web menghasilkan dokumen secara lokal untuk

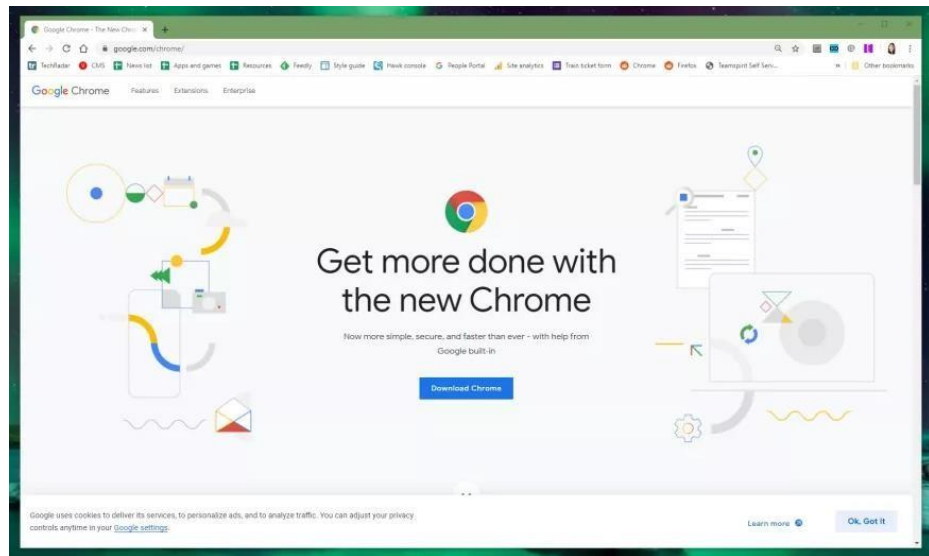
digunakan secara *offline*.

Seringkali penyimpanan sisi-*client* dan sisi-*server* digunakan bersama-sama. Misalnya, Anda dapat mengunduh dari basis data sejumlah file musik yang digunakan oleh permainan web atau aplikasi pemutar musik menyimpannya di dalam basis data sisi *client*, dan memutarnya sesuai kebutuhan. Pengguna hanya perlu mengunduh file musik satu kali - pada kunjungan berikutnya mereka akan diambil dari *database* sebagai gantinya (Camden, 2015).

2.3. Browser

Peramban web adalah bagian terpenting dari peranti lunak yang harus dikerjakan oleh perancang web. Ini adalah program yang Anda gunakan untuk melihat halaman dan menavigasi web. Tujuan utama dari *browser* web adalah untuk terhubung ke *server* web, meminta dokumen, dan kemudian memformat dan menampilkan dokumen-dokumen tersebut dengan benar. Peramban web juga dapat menampilkan file di komputer lokal Anda, mengunduh file yang tidak dimaksudkan untuk ditampilkan, mengirim/menerima *email*, dan tergantung pada fitur yang diaktifkan, dapat memainkan animasi *flash*, membuka file *pdf*, atau menjalankan *applet Java*. Apa yang terbaik dari *browser*, bagaimanapun, adalah berurusan dengan mengambil dan menampilkan dokumen web. *Browser* yang berbeda mungkin memformat dan menampilkan file yang sama dengan cara yang berbeda, tergantung pada kemampuan sistem dan bagaimana *browser* dikonfigurasi

2.3.1 Google Chrome



Gambar 2. 4 Browser google chrome

Sebagian besar pengguna web sangat terbiasa dengan *browser* milik raksasa pencarian *Google*, yakni *Google Chrome*. *Browser* ini memiliki desain yang menarik dan pemuatan halaman yang cepat. Sebagian besar kode situs web sekarang menargetkannya, jadi kompatibilitas jarang menjadi masalah. Meskipun demikian, setiap *browser* terkadang bingung dengan satu atau dua situs tertentu, dan terkadang pembaruan *browser* bahkan dapat merusak situs yang telah dibuat dengan baik. *Chrome* mendapat nilai tertinggi di situs web *HTML5Test*. Ini juga merupakan tolok ukur untuk *JetStream 2*, yang menguji kecepatan berbagai teknologi web canggih. *Chrome* menggunakan lebih banyak RAM daripada *browser windows* lainnya, tetapi beberapa diantaranya untuk mempercepat operasi dengan memuat konten terlebih dahulu. *Google* selalu berkomitmen pada

keamanan dan peningkatan fungsional, tetapi seperti semua perangkat lunak, kesalahan dapat terjadi, jadi pastikan untuk selalu mendapatkan informasi terbaru.

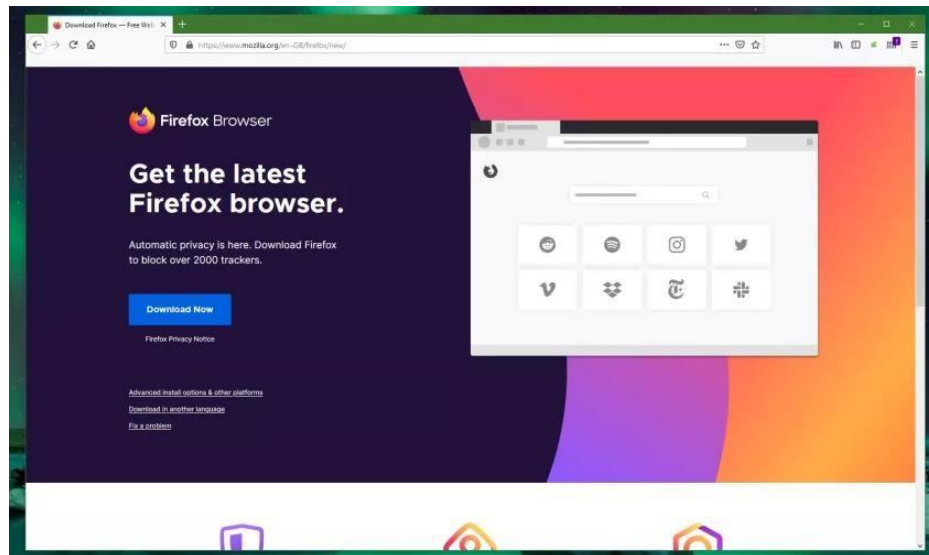
Chrome dikenal dengan kebutuhan sumber dayanya yang besar dan cukup dapat digunakan pada perangkat keras berdaya rendah dengan RAM terbatas. Fitur pembekuan tab baru bertujuan untuk memecahkan masalah ini dengan secara otomatis "membekukan" tab latar belakang sehingga mereka tidak menggunakan sumber daya yang tidak perlu, tetapi *Chrome* masih membutuhkan banyak perangkat keras.

Versi *Chrome* saat ini memenuhi lebih banyak standar web daripada *browser* lain, dan siklus pembaruan yang sering berarti bahwa masalah keamanan dan kesalahan lainnya diatasi dengan cepat. *Chrome* juga melakukan pekerjaan yang luar biasa dalam mengelola tab. Selain mengubah ukuran tab secara cerdas dan menyediakan opsi menyematkan untuk akses mudah, *browser* juga menyimpan tab individual dalam memori secara terpisah (Marshall, 2021).

2.3.2 Mozilla Firefox

Mozilla Firefox baru-baru ini menerima pembaruan terbesarnya dalam 13 tahun, memecahkan banyak masalah yang menyebabkan penurunan peringkatnya dalam beberapa tahun terakhir. *Firefox* telah lama dikenal karena fleksibilitas dan dukungannya untuk ekstensi, tetapi dalam beberapa tahun terakhir, *Firefox* mulai tertinggal dari pesaingnya dalam hal

kecepatan, dan banyak pengguna beralih ke *Chrome*. Versi terbaru berkinerja baik dalam uji kecepatan, dan *browser* disederhanakan, jelas hal ini dipengaruhi oleh *Google Chrome* yang populer.



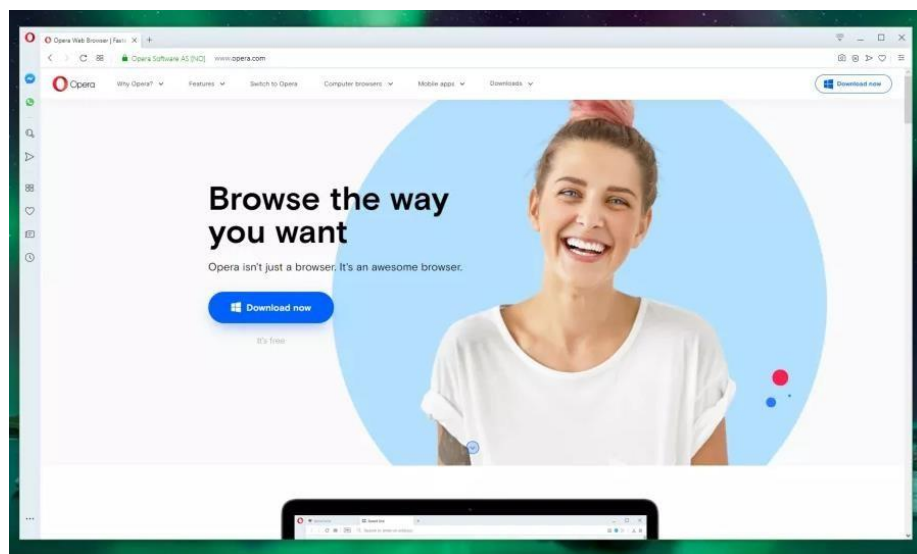
Gambar 2. 5 Browser mozilla firefox

Setelah beberapa tahun tertinggal dari pesaing utamanya, Chrome dan Edge, Firefox telah kini telah hidup kembali, memulai rilis versi Quantum dimana pembaruan terbesar browser ini sejak diluncurkan 14 tahun yang lalu. Salah satu yang terbaik adalah alat tangkapan layar (dapat diakses melalui menu konteks klik kanan), yang menangkap seluruh halaman web tanpa perlu menggulir, atau hanya bagian yang diinginkan. Tangkapan layar kemudian akan diunggah ke server Mozilla sehingga Anda dapat membagikan atau mengunduhnya selama 14 hari.

Quantum mewakili perubahan besar di Firefox, tetapi perubahan kecil ini juga akan berdampak nyata pada pengalaman browsing dan menjadikannya salah satu browser terbaik saat ini. Antarmuka telah

sepenuhnya didesain ulang, dengan tampilan yang lebih jelas dan animasi yang lebih halus, membantu meningkatkan peningkatan kinerja yang terukur. Semua fitur keamanan browser sekarang dapat ditemukan di menu opsi utama, alih-alih tersebar di berbagai bagian antarmuka. Perlindungan phishing dan pemblokiran pop-up diaktifkan secara default, dan perlindungan pelacakan dapat dilakukan hanya dengan satu klik (Marshall, 2021).

2.3.3 Opera Browser



Gambar 2. 6 Browser opera

Opera memiliki pilihan fitur yang unik, menjadikannya kandidat yang sangat berharga dalam salah satu *browser* terbaik 2020. Faktanya, Opera dapat membuat pernyataan bahwa privasi yang lebih besar daripada browser lain mana pun, jika percaya pada VPN, karena ini menyertakan VPN bawaan yang berfungsi dengan baik dan cepat. Beberapa orang berpikir

bahwa VPN Opera adalah server proxy terenkripsi yang nyata, tetapi satu-satunya perbedaan nyata antara VPN ini dan VPN standar adalah VPN ini hanya melindungi dan mentransfer lalu lintas dari Opera, bukan dari aplikasi yang terhubung ke internet di komputer atau ponsel cerdas.

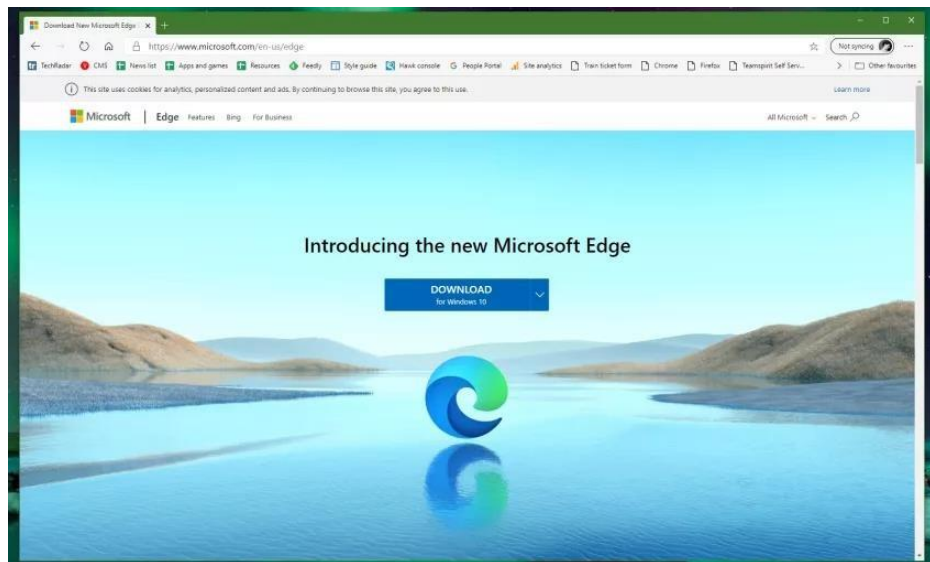
Opera menggunakan mesin *rendering* halaman *Chromium*, sehingga jarang menemukan ketidakcocokan situs dan kinerjanya sangat cepat. *Opera* juga menggunakan lebih sedikit ruang *disk* dan memori daripada *Chrome*.

Opera dikemas dengan fitur yang berguna, tetapi salah satu yang favorit bagi penggunanya tidak lagi ada di *browser* desktop: *Opera Turbo*, yang mengompres data internet seperti gambar sehingga hal-hal dimuat lebih cepat pada koneksi omong kosong, sekarang hanya tersedia untuk *browser* seluler. Namun, tersedia fitur mode hemat baterai yang berguna ketika unduhan yang dilakukan lambat setidaknya tidak perlu khawatir tentang baterai laptop yang akan terkuras (Marshall, 2021).

2.3.4 Microsoft Edge

Microsoft pernah menjadi raja penjelajahan web dengan browser Internet Explorer-nya yang sekarang usang mencapai 95% pangsa pasar pada tahun 2003, sebagian besar berkat pra-instalasi pada semua mesin Windows baru. Ketikapesaing datang dengan browser yang lebih cepat dan lebih menarik, menyelinap secara diam-diam, hingga pada 2015 merilis Microsoft Edge untuk Windows 10 dan Xbox One. Awal tahun ini, ketika

Microsoft bermitra dengan Google untuk merilis versi baru Edge, itu mengalami kebangkitan. Di mesin Chromium.



Gambar 2. 7 Browser microsoft edge

Microsoft adalah pakar pengalaman pengguna jangka panjang dan membawa pengalaman pengembangan sistem operasi selama puluhan tahun ke dalam pengembangan browser terbarunya, seperti penggulingan yang lebih mulus dan navigasi yang lebih cepat pada sistem yang lebih lama. Versi baru Microsoft Edgetidak hanya untuk pengguna PC, tetapi juga mendukung Android, iOS, dan macOS. Edge adalah browser default di Windows 10, dengan pengaturan privasi dasar, seimbang, dan ketat serta beranda yang dapat disesuaikan. Anda dapat melihat situsweb yang paling sering dikunjungi di halaman kosong dan menikmati foto latarbelakang baru setiap hari. Anda juga bisa mendapatkan informasi berita, cuaca, olahraga, dan keuangan yang disesuaikan untuk Anda.

Dengan sidebar favorit, Anda dapat dengan cepat menarik dan melepas halaman web, gambar, dan catatan ke dalam Microsoft Excel atau Microsoft Word. Jika Anda telah menggunakan perangkat Windows untuk sementara waktu, Microsoft Edge mungkin menjadi kebiasaan Anda, dan karena dirancang untuk mereka, itu adalah pilihan yang baik (Marshall, 2021).

2.4. HTML

Sejarah *HTML* bermula ketika Tim Berners-Lee dan peneliti lain mencari solusi dari masalah aksesibilitas informasi dan berbagi di fasilitas penelitian nuklir CERN, Jenewa, Swiss pada tahun 1989. Mereka memperkenalkan *hypertext* yang dapat memungkinkan koneksi di antara berbagai sumber dengan *hyperlink*. Setelah beberapa saat, metode ini menjadi populer di kalangan peneliti lain di dunia dan seluruh koneksi diantara halaman-halaman ini di internet dinamakan sebagai *World Wide Web*. Struktur *HTML* didasarkan pada elemen yang diwakili oleh tag (Maennel, 2018).

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Gambar 2. 8 Dokumen html sederhana

Pada gambar di atas, `<HTML>`, `<head>`, `<title>`, dan `<head>` adalah tag yang biasanya muncul berpasangan. Sejak penemuan WEB, ada berbagai versi *HTML*:

Year	Version
1989	Tim Berners-Lee invented www
1991	Tim Berners-Lee invented HTML
1993	Dave Raggett drafted HTML+
1995	HTML Working Group defined HTML 2.0
1997	W3C Recommendation: HTML 3.2
1999	W3C Recommendation: HTML 4.01
2000	W3C Recommendation: XHTML 1.0
2008	WHATWG HTML5 First Public Draft
2012	WHATWG HTML5 Living Standard
2014	W3C Recommendation: HTML5
2016	W3C Candidate Recommendation: HTML 5.1

Tabel 2. 1 Versi html

2.5. *Database*

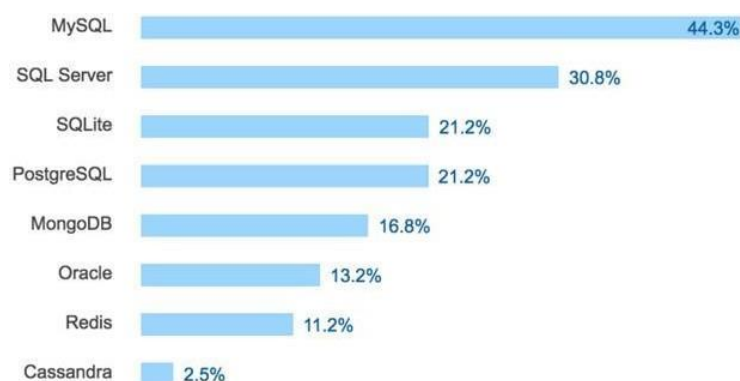
Database adalah kumpulan data. Secara umum, ada berbagai bentuk *database* yang tersedia seperti *database* rasional, objek, dan *database* objek-rasional, dan lain-lain. Berdasarkan penggunaan dan jenis data, kita perlu menggunakan *database* yang berbeda. Penelitian ini tidak mampu membahas semuanya. Silberschatz memberikan gambaran komprehensif tentang konsep *database* (Maennel, 2018).

Namun, sebelum melangkah lebih jauh, kita perlu memahami dua jenis utama *database* pada umumnya: *SQL* dan *NoSQL*.

2.5.1 SQL dan NoSQL database

SQL atau bahasa *query* terstruktur adalah bahasa pemrograman yang telah dirancang untuk bekerja dengan sistem manajemen basis data relasional atau *RDBMS*. Struktur *SQL* terdiri dari tabel dan tabel dibagi menjadi baris dan kolom.

NoSQL atau *Non-SQL* adalah *database* yang memungkinkan penyimpanan dan pengambilan data tidak hanya dengan tabel, itu juga bisa merujuk ke tidak hanya *SQL* yang berarti mungkin memiliki pertanyaan serupa yang digunakan dalam *SQL*. *NoSQL* memainkan peran utama dalam sistem *database* ketika skalabilitas tinggi dan bekerja dengan data besar. Basis data *NoSQL* menggunakan metode yang berbeda untuk menyimpan dan mengambil data. Salah satunya adalah pasangan kunci / nilai. Itu tidak terstruktur, tidak rasional dan tidak berorientasi objek. Kuncinya adalah atribut untuk data, dan nilainya bisa berupa data atau penunjuk ke file apa pun atau alamatnya. Berdasarkan survei Stack Overflow 2018 tentang *database*, *RDBMS* masih yang paling umum daripada *database NoSQL* seperti *MongoDB* (Maennel, 2018).



Gambar 2. 9 Grafik survei penggunaan database

Namun, *database NoSQL* muncul secara signifikan karena mereka dapat memberikan kinerja yang mengesankan dalam kecepatan dan ukuran. "Ketersediaan tinggi" ini memiliki efek buruk pada *ACID (Atomic, Consistent, Isolated, Durable)* dari *database*. Selain itu, ketika transaksi besar diperlukan mengingat perusahaan-perusahaan seperti *Google, Amazon, Microsoft, dan Facebook*, itu juga merupakan pilihan pertama untuk menyimpan data. Cocok untuk perusahaan kecil dan perusahaan tanpa kekhawatiran untuk desain dan jenis data yang akan disimpan adalah poin menarik lainnya untuk menggunakan *NoSQL* alih-alih *database SQL*.

Beberapa basis data *NoSQL* yang populer adalah *MongoDB, Google's Big Table, CouchDB, dan Cassandra*. Mempertimbangkan kelemahan dari *database NoSQL* lebih dari yang rasional, tidak ada ketersediaan bahasa *query* standar, "tidak ada antarmuka standar" dan pemeliharaan yang menantang harus disebutkan (Maennel, 2018).

2.5.2 Server-side dan client-side database

Basis data web dibagi menjadi dua kategori utama: sisi *server* dan sisi *client*. Sisi *server* yang memproses dan menyimpan data terjadi di *server*, dan sisi *client* adalah sebaliknya, artinya terjadi pada mesin pengguna. Cara lain untuk mengklasifikasikan basis data web adalah *online* (sebagian besar sebagai sisi *server*) dan *offline* (sering sebagai sisi *client*). Yang pertama membutuhkan koneksi ke internet, dan yang terakhir dapat diakses tanpa koneksi apapun (Maennel, 2018).

2.5.3 Tinjauan umum tentang *database sisi client*

Seperti disebutkan sebelumnya, berdasarkan penggunaan dan tipe data kita dapat memilih basis data mana yang akan digunakan. Ada banyak alasan untuk mempertimbangkan basis data sisi *client* seperti:

- Lebih responsif daripada sisi *server*. Akan lebih cepat untuk mengakses dan menyimpan data secara lokal. Oleh karena itu sangat membantu untuk meningkatkan kinerja dibandingkan dengan sisi *server*.
- Dapat digunakan dalam mode *offline*. Ini juga akan membuat aplikasi web berguna ketika ada koneksi lemah, aksesibilitas dan peningkatan ketersediaan.
- Mengurangi beban *server* serta penggunaan *bandwidth*.
- Memberi kesempatan untuk menyimpan aktivitas pengguna sebelumnya di situs web dan penyesuaian situs web seperti *widget* dan skema warna atau ukuran *font*.

HTTP Cookie adalah basis data sisi *client* yang paling populer. Itu menggunakan sesi *stateful* untuk menghubungkan *server* dan *client* untuk kebutuhan mereka. Setiap *cookie* adalah 4 KB, dan disimpan di *browser* pengguna. Meskipun, ini adalah mekanisme yang sederhana dan didukung dengan baik; kurangnya persyaratan privasi dan keamanan dasar. Komisi Eropa telah menetapkan beberapa dasar tentang cara menggunakan *cookie*.

Keracunan *cookie* dan injeksi *cookie* adalah beberapa contoh masalah keamanan. Yang pertama adalah jenis serangan yang penyerang memanipulasi konten *cookie* dan memotong sistem keamanan. Yang terakhir terjadi jika penyerang dapat menyuntikkan kode atau string ke header *HTTP* untuk menjalankan perintah di *server*.

Produktivitas dan fleksibilitas terlalu lemah dibandingkan dengan pesaing lain. Ini juga memiliki *overhead* tambahan karena harus bereaksi dengan setiap permintaan *HTTP* antara *server* dan *client*.

Cookie flash adalah jenis *cookie* yang tidak lagi digunakan karena pemasangan plugin. Seperti yang disebutkan di bagian pendahuluan, sebagian besar *browser* sudah berhenti untuk mendukung *plugin flash* atau hanya memberikan akses terbatas kepada mereka.

Google Gears adalah jenis lain yang menghentikan pengembangannya untuk lebih fokus pada API *HTML5* seperti *IndexedDB*, File API.

Web SQL Database tidak lagi dalam pemeliharaan oleh Kelompok Kerja Aplikasi Web, dan mereka merekomendasikan menggunakan Web Storage dan *IndexedDB*.

JavaScript Variabel dianggap sebagai pilihan paling sederhana untuk menyimpan data di sisi *client*. Namun, ia tidak memiliki cukup fitur dengan terlalu banyak keterbatasan untuk dipertimbangkan sebagai *database* yang tepat. Kisah ini juga berlaku untuk properti *windows.name*.

File API saat ini adalah W3C Working Draft. Ini menyajikan antarmuka File yang “menyediakan informasi tentang file dan

memungkinkan JavaScript di halaman web untuk mengakses konten mereka. Antarmuka Blob (Binary Large Object) adalah data mentah yang tidak berubah. File dan Blob keduanya harus terjadi secara tidak sinkron; ini mencegah aplikasi web memblokir dan membeku.

Web storage API adalah rekomendasi W3C yang mewakili dua mekanisme serupa dengan *Cookie* sesi HTTP. `sessionStorage` menyimpan data hanya untuk satu sesi, dan setelah menutup tab *browser*, data akan hilang (Maennel, 2018).

2.6. DexieJS

Dexie.js adalah pustaka pembungkus untuk *IndexedDB* - database standar di browser. Dexie JS adalah sebuah library JavaScript yang berguna untuk membuat *database* tanpa memerlukan suatu *server*. Dengan DexieJS, hanya memerlukan *browser* sebagai tempat penyimpanan *databasenya*. Dexie menggunakan teknologi *IndexedDB*, yaitu suatu *database* manager di *browser* untuk menyimpan data. Dexie.js membungkus *IndexedDB* secara minimalis sehingga memberikan kinerja yang hampir sama dan basis data yang mudah digunakan. Dexie memecahkan tiga masalah utama API *IndexedDB* asli:

- Penanganan kesalahan ambiguitas
- Kueri yang buruk
- Kompleksitas kode

Dexie menyediakan API basis data yang ringkas dengan desain API yang dipikirkan dengan baik, penanganan kesalahan yang kuat, skalabilitas, pengetahuan tentang pelacakan perubahan, dan dukungan KeyRange yang diperluas (Dexie.js - Minimalistic IndexedDB Wrapper, 2014).

2.7. *IndexedDB*

IndexedDB adalah kandidat yang direkomendasikan paling menarik di antara semua *database*, ketika datang ke data skala besar, tanpa batasan. Ini memungkinkan pengembang untuk membangun aplikasi web dengan kemampuan permintaan yang kuat. *IndexedDB* cocok untuk menyimpan sejumlah besar data seperti laboratorium *online*, perpustakaan, atau penggunaan aplikasi web apa pun tanpa perlu koneksi internet permanen seperti widget, daftar agenda.

IndexedDB menggunakan banyak transaksi untuk menyimpan dan mengambil data. Ini adalah *database* JavaScript berorientasi objek. Objek yang didukung oleh *IndexedDB* adalah semua objek algoritma tertutup terstruktur termasuk semua tipe primitif, objek Boolean, objek String, Tanggal, Blob, File, FileList, ImageData, Array, ObjectMap, dll. Algoritma clone terstruktur adalah algoritma yang didefinisikan oleh Spesifikasi *HTML5* untuk menyalin objek JavaScript kompleks. Selain itu efisiensi menyimpan data di *IndexedDB* sama dengan menyimpan file di OS. Pola dasar *IndexedDB* adalah:

1. *Open database.*
2. Membuat sebuah *Object Store* di *database*.
3. Mulai transaksi dan minta operasi basis data, seperti menambah

atau mengambil.

4. Menunggu selesainya operasi yang ditangani oleh DOM.
5. Menampilkan hasilnya.

Pertama, kita perlu menambahkan awalan berikut:

```
var indexedDB = window.indexedDB ||  
window.mozIndexedDB || window.webkitIndexedDB ||  
window.msIndexedDB;  
  
//prefixes of window.IDB objects  
var IDBTransaction = window.IDBTransaction ||  
window.webkitIDBTransaction ||  
window.msIDBTransaction;  
var IDBKeyRange = window.IDBKeyRange ||  
window.webkitIDBKeyRange || window.msIDBKeyRange  
  
if (!indexedDB) {  
    alert("Your browser does'nt support a stable  
version of IndexedDB.")  
}
```

Gambar 2. 10 Implementasi IndexedDB

```
var userData = [  
    { id: "1", name: "Tapas", age: 33, email:  
"tapas@example.com" },  
    { id: "2", name: "Bidulata", age: 55, email:  
"bidu@home.com" }  
];
```

Gambar 2. 11 Sampel data

Kemudian *Open database*, dan setelah itu buat dan tambahkan data ke tabel:

```
var db;
var request = indexedDB.open("databaseName", 1);

request.onerror = function(e) {
    console.log("error: ", e);
};

request.onsuccess = function(e) {
    db = request.result;
    console.log("success: " + db);
};

request.onupgradeneeded = function(e) {
}

request.onupgradeneeded = function(event) {
    var objectStore =
event.target.result.createObjectStore("users",
{keyPath: "id"});
    for (var i in userData) {
        objectStore.add(userData[i]);
    }
}
```

Gambar 2. 12 Open DB, create, tambah tabel

Dukungan tiap versi browser pada *IndexedDB* dapat ditemukan dalam rincian referensi berikut:

IndexedDB  - REC

Method of storing data client-side, allows indexed database queries.

Current aligned Usage relative Date relative Filtered All 

IE	Edge [*]	Firefox	Chrome	Safari	Opera
		2-3.6	4-10		
		4-9 	11-22 	3.1-7	
6-9	¹ 12-18	10-15 	23 	² 7.1-9.1	10-12.1
¹ 10	79-91	16-89	24-91	10-14	15-76
¹ 11	92	90	92	³ 14.1	77
		91-92	93-95	15-TP	

Gambar 2. 13 Browser support

Berikut beberapa masalah yang diketahui dari versi-versi web browser tersebut:

- Dukungan parsial di IE 10 dan 11 mengacu pada beberapa sub-fitur yang tidak didukung. Edge tidak mendukung *IndexedDB* dalam *Blob* *Web Workers*.
- Bagian dari dukungan di Safari dan iOS 8 dan 9 mengacu pada perilaku buruk yang serius dan kurangnya dukungan di WebViews.
- Dukungan parsial mengacu pada bug di versi 14.1.1 yang menyebabkan *IndexedDB* tidak pernah dimuat saat browser pertama kali dibuka.
- Firefox (sebelum versi 37) dan Safari tidak mendukung *IndexedDB* di

Web Workers.

- Firefox dan Edge tidak mendukung *IndexedDB* dalam mode penjelajahan pribadi.
- Chrome 47 untuk iOS dan versi yang lebih rendah atau WebView iOS lama lainnya yang menggunakan UIWebView alih-alih WKWebView tidak mendukung. Chrome 36 dan versi yang lebih rendah tidak mendukung objek Blob sebagai nilai *IndexedDB*.
- Chrome memiliki bug di mana *IndexedDB* yang dihapus dapat dibuat ulang saat alat dev terbuka (diperbaiki di build terbaru).
- Microsoft Edge tidak mendukung *IndexedDB* dalam Blob Web Workers.

IndexedDB, sebelumnya dikenal sebagai WebSimpleDB, berasal dari spesifikasi W3C untuk mengimplementasikan penyimpanan Web di browser Web pada tahun 2009. *IndexedDB* adalah database persisten sisi klien yang diimplementasikan di browser. Ini adalah alternatif untuk WebSQL dan kemudian ditolak. Mozilla dan Microsoft mendukung pengenalan *IndexedDB*, yang paling terpengaruh oleh Oracle Berkley DB. Aplikasi ini menggunakan data lokal yang tersimpan di sistem klien. Ini menggunakan penyimpanan objek JavaScript untuk menyimpan sejumlah besar data dari server ke klien browser Web, yang dapat dianggap sebagai setara dengan tabel dalam database relasional.

File dan data yang disimpan oleh browser disimpan dalam sistem penyimpanan file pengguna, yang terletak di hard drive komputer pengguna. Database klien *IndexedDB* menyimpan data bahkan jika browser dihentikan.

IndexedDB adalah database klien persisten, yang berarti bahwa data dapat diambil bahkan saat browser offline. Oleh karena itu, file berada di sistem file pengguna dan dapat dipulihkan hingga ditimpa oleh file lain. *IndexedDB* memperlakukan datafile dengan cara yang sama seperti tipe data lainnya. Aplikasi dapat menulis file atau blob ke *IndexedDB*, serta menyimpan string, angka, dan objek JavaScript. Inidirinci dalam spesifikasi *IndexedDB*, yang saat ini diterapkan di aplikasi *IndexedDB* Firefox dan Chrome. Dengan ini, semua informasi disimpan di satu tempat, dan kueri ke *IndexedDB* dapat mengembalikan semua data.

Dalam implementasi *IndexedDB* Firefox dan Chrome, file disimpan secara transparan di luar database yang sebenarnya; dengan kata lain, kinerja menyimpan beberapa data di *IndexedDB* sama efektifnya dengan menyimpannya langsung di sistem file sistem operasi. Menyimpan file tidak akan menambah ukuran database dan akan memperlambat operasi lainnya. Juga, membaca dari file berarti membaca implementasi file dari sistem operasi. Implementasi Firefox *IndexedDB* bahkan cukup pintar untuk mengenali jika Blob yang sama disimpan dalam banyak file. Jika ini terjadi, itu hanya akan membuat satu salinan. Menulis lebih banyak referensi ke Blob yang sama hanya akan meningkatkan penghitungan referensi internal. Ini benar-benar transparan untuk halaman web, sehingga kecepatan penulisan data lebih cepat dan menggunakan lebih sedikit sumber daya.

IndexedDB didasarkan pada model database transaksional. Ini adalah fitur hebat dari *IndexedDB*. Selama *IndexedDB* adalah database lokal yang berjalan di browser. Sebagian besar browser saat ini mendukung banyak tab; membuka contoh

database lain dari tab lain tanpa manajemen transaksi dapat menyebabkan aplikasi mogok. Keamanan dikelola oleh kebijakan asal yang sama. Kebijakan asal yang sama adalah model keamanan yang digunakan untuk melindungi aplikasi Web. Dalam model ini, cookie, JavaScript, dan *IndexedDB* terbaru dari satu situs tidak dapat berinteraksi satu sama lain, meskipun semuanya berada di browser yang sama pada mesin yang sama. Basis data dilindungi oleh kebijakan asal yang sama, yang mengontrol akses ke domain, protokol lapisan aplikasi, dan port yang sama.

Setiap browser memiliki implementasi *IndexedDB* sendiri, misalnya Internet Explorer menggunakan Extensible Storage Engine, Firefox menggunakan SQLite, dan Chrome menggunakan LevelDB.

- Tambahan: Objek JSON vs Objek Javascript

Apa bedanya objek pada JSON dengan objek pada Javascript karena jika diperhatikan bentuk dan strukturnya sama persis. Keduanya sama persis, tetapi tipe keduanya berbeda, JSON adalah teks biasa, sehingga objek pada JSON juga merupakan string (teks) dan karena JSON mengadopsi ECMAScript – Javascript maka format yang digunakan juga akan sama.

Saat ini JSON menjadi format standar untuk pertukaran data dan lebih banyak digunakan untuk pertukaran data via internet (protokol HTTP) karena load time yang lebih cepat dan bandwidth yang diperlukan lebih kecil. Dengan struktur yang simpel, format JSON lebih mudah digunakan dibanding format yang lain, terutama jika data yang di pertukarkan jumlahnya besar. JSON adalah format pertukaran data, yang

kebetulan terlihat seperti bagian dari kode YAML atau JavaScript yang dapat Anda jalankan dan dapatkan objek kembali. Objek JavaScript hanyalah objek dalam JavaScript. Dengan JSON sebagai format pertukaran data, Anda dapat bertukar data terstruktur dalam bentuk tekstual dengannya. Ini cukup dipisahkan dari JavaScript sekarang. Objek JavaScript memungkinkan Anda membuat dan bekerja dengan data terstruktur selama pelaksanaan program JavaScript.