BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Dalam penelitian ini terdapat beberapa tinjauan pustaka yang digunakan sebagai bahan acuan adalah sebagai berikut :

Tabel 2.1 Tinjauan Pustaka.

Peneliti, Tahun	Objek / Data	Metode	Teknologi	Hasil
Muhamad Mulya Fuadi Agisna[1], Bagus Dwi Sandi Putra[2], 2013	*Database engine* Microsoft SQL Server 2008	*Data Access Objects*	Web	Aplikasi Web berorientasi objek dengan bahasa pemrograman *Java*
Darlis Heru 0Murti[1], Yudhi Purnawanto[2], M. Rifqi Febianto[3], 2004	RDBMS Oracle	*Query Builder*	Web	Aplikasi Web *Client-Server* dengan mengimplementasikan *Query Builder*
Badiyanto, 2016	Sistem Informasi Akademik, Keuangan, Kepegawaian di STMI AKAKOM Yogyakarta.	*Data Access Object, Query Builder, CActiveRecord,* php *native.*	Web	Analisis dukungan relasi tabel antar database dari masing-masing metode.
Ahmad Zakir, 2017	*Yii Framework*	*Yii Framework, Data Access Object, Query Builder, CActiveRecord.*	Web	Aplikasi Web dengan menggunakan *Yii Framework.*
Usva Dhiar Praditya1, Ragil Saputra2, Beta Noranita3, 2013	*E-commerce*	*CActiveRecord*	Web	Aplikasi Web *E-commerce* dengan *Yii Framework*

Skripsi dengan judul "Analisis Perbandingan Kecepatan Metode *CActiveRecord, Query Builder,* dan *Data Access Object* pada *Framework Yii*" ini, menggunakan kelima penelitian tersebut sebagai acuan. Penelitian ini menggunakan metode *CActiveRecord, Query Builder,* dan *Data Access Object* yang akan diimplementasikan ke dalam Aplikasi berbasis Web sehingga mampu membaca data pada database.

2.2 Dasar Teori

Pada bab ini dibahas mengenai dasar teori yang akan digunakan dalam pembahasan selanjutnya yaitu teori-teori *CActiveRecord, Query Builder*, *Data Access Object*, *Query Benchmark,* dan Diagram.

2.2.1 CActiveRecord

Active Record (AR) adalah teknik populer Pemetaan Relasional Objek / Object-Relational Mapping (ORM). Setiap kelas AR mewakili tabel database (atau view) yang atributnya diwakili oleh properti kelas AR, dan turunan AR mewakili baris pada tabel tersebut. Operasi umum CRUD diimplementasikan sebagai metode AR. Hasilnya, kita dapat mengakses data dengan cara lebih terorientasi-objek. Sebagai contoh, kita dapat menggunakan kode berikut untuk menyisipkan baris baru ke dalam tabel Post:

```
$post=new Post;
$post->title='sample_post';
$post->content='post body content';
$post->save();
```

Selanjutnya kita mendalami bagaimana menyiapkan AR dan menggunakannya untuk melakukan operasi CRUD. Kita akan melihat bagaimana

untuk menggunakan AR untuk bekerja dengan relasi database dalam seksi berikutnya. Demi kemudahan, kami menggunakan tabel database berikut sebagai contoh kita dalam bagian ini. Harap dicatat bahwa jika Anda menggunakan database MySQL, Anda harus mengganti AUTOINCREMENT dengan AUTO_INCREMENT dalam SQL berikut.

```
CREATE TABLE tbl_post(
      id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
      title VARCHAR(128) NOT NULL,
      content TEXT NOT NULL,
      create_time INTEGER NOT NULL
);
```

1. Membuat koneksi Database

AR bergantung pada koneksi DB untuk melaksanakan operasi terkait-DB. Secara default, AR menganggap bahwa komponen aplikasi db adalah turunan CDbConnection yang dibutuhkan untuk bertindak sebagai koneksi DB. Konfigurasi aplikasi berikut memperlihatkan sebuah contoh:

```
Return array(
   'components'=>array(
        'db'=array(
             'class'=>'system.db.CBbConnection';
                'connectionString'=>'sqlite:path/to/dbfile',
         ),
      ),
);
```

Dukungan terhadap AR dibatasi oleh DBMS. Saat ini, hanya DBMS berikut yang didukung:

- MySQL 4.1 atau lebih tinggi
- PostgreSQL 7.3 atau lebih tinggi
- SQLite 2 dan 3
- Microsoft SQL Server 2000 atau lebih tinggi
- Oracle

2. Mendefinisikan kelas AR

Untuk mengakses tabel database, pertama kita perlu mendefinisikan kelas AR dengan menurun CActiveRecord. Setiap kelas AR mewakili satu tabel database, dan instance AR mewakili sebuah record (baris) dalam tabel tersebut. Contoh berikut memperlihatkan kode minimal yang diperlukan untuk kelas AR yang mewakili tabel Post

```
class Post extends CActiveRecord
{
    public static function model($className=__CLASS__)
    {
        return parent::model($className);
    }

    public function tableName()
    {
        return 'tbl_post';
    }
}
```

Secara default, nama kelas AR sama dengan nama tabel database. Meng-override metode tableName() jika berbeda. Metode model() dideklarasikan seperti itu untuk setiap kelas AR (akan dijelaskan kemudian). Nilai kolom pada baris tabel dapat diakses sebagai properti turunan kelas AR terkait. Sebagai contoh, kode berikut menyetel kolom title (atribut):

```
$post=new Post;
$post->title='a sample post';
```

AR bergantung pada pendefinisian primary key tabel yang baik. Jika sebuah tabel tidak memiliki primary key, maka AR membutuhkan kelas AR menentukan kolom mana yang dijadikan primary key dengan meng-overide fungsi primaryKey() sebagai berikut,

```
public function primaryKey()
{
    return 'id';
    // Kalau composite primary key, mak kembalikan nilai
array seperti demikian
    // return array('pk1', 'pk2');
}
```

3. Membuat record

Untuk melakukan insert baris baru ke dalam tabel database, kita membuat instance baru dari kelas AR terkait, menyetel propertinya yang berkaitan dengan kolom tabel, dan memanggil metode save() untuk menyelesaikan proses insert.

```
$post=new Post;
$post->title='sample post';
$post->content='content for the sample post';
$post->createTime=time();
$post->save();
```

Jika primary key table bersifat auto-increment, setelah insert instance AR maka akan berisi primary key yang baru. Dalam contoh di atas, properti id akan merujuk pada nilai primary key tulisan yang baru saja disisipkan, meskipun kita tidak pernah mengubahnya secara eksplisit.

Jika kolom didefinisikan dengan beberapa nilai standar statis (misalnya string, angka) dalam skema tabel, properti terkait dalam instance AR akan secara otomatis memiliki nilai tersebut setelah instance dibuat. Satu cara untuk mengubah nilai default ini adalah dengan secara eksplisit mendeklarasikan properti dalam kelas AR:

```
class Post extends CActiveRecord
{
    public $title='please enter a title';
    ......
}
$post=new Post;
echo $post->title;  // ini akan menampilkan: please enter a
title
```

Atribut dapat menempatkan nilai tipe CDbExpression sebelum record disimpan (baik skenario insert ataupun update) ke database. Sebagai contoh, untuk menyimpan timestamp yang dihasilkan oleh fungsi MySQL NOW(), kita dapat menggunakan kode berikut:

```
$post=new Post;
$post->createTime=new CDbExpression('NOW()');
// $post->createTime='NOW()'; tidak akan bekerja karena
// 'NOW()' akan dianggap sebagai string
$post->save();
```

4. Membaca record

Untuk membaca data dalam tabel database, kita memanggil salah satu metode find seperti berikut.

```
// cari baris pertama sesuai dengan kondisi yang ditetapkan
$post=Post::model()->find($condition,$params);
// cari baris dengan primary key yang ditetapkan
$post=Post::model()->findByPk($postID,$condition,$params);
// cari baris dengan nilai atribut yang ditetapkan
$post=Post::model()-
>findByAttributes($attributes,$condition,$params);
// cari baris pertama menggunakan pernyataan SQL yang ditetapkan
$post=Post::model()->findBySql($sql,$params);
```

Dalam contoh di atas, kita memanggil metode find dengan Post::model(). Ingat bahwa metode statis model() diperlukan oleh setiap kelas AR. Metode ini menghasilkan instance AR yang dipakai untuk mengakses metode tingkat kelas (mirip dengan metode kelas static) dalam konteks obyek.

Jika metode find menemukan baris yang sesuai dengan kondisi query, ia akan mengembalikan turunan Post yang propertinya berisi nilai kolom terkait dari baris table. Kemudian kita dapat membaca nilai yang diambil sepert yang kita lakukan pada properti obyek, sebagai contoh, echo $post->title;.

Metode find akan menghasilkan null jika tidak ada yang ditemukan dalam database dengan kondisi query yang diberikan.

Ketika memanggil find, kita menggunakan $condition dan $params untuk menetapkan kondisi query. Di sini, $condition dapat berupa string yang mewakili klausul WHERE dalam pernyataan SQL, dan $params adalah array parameter yang nilainya harus diikat ke tempat di dalam $condition. Sebagai contoh,

```
// cari baris dengan postID=10
$post=Post::model()->find('postID=:postID',
array(':postID'=>10));
```

Kita juga bisa menggunakan $condition untuk menetapkan kondisi query lebih kompleks Alih-alih mengisi sebuah string, kita dapat mengatur $condition menjadi instance CDbCriteria yang mengijinkan kita untuk menetapkan kondisi selain klausul WHERE. Sebagai contoh,

```
$criteria=new CDbCriteria;
$criteria->select='title';  // hanya memilih kolom 'title'
$criteria->condition='postID=:postID';
$criteria->params=array(':postID'=>10);
$post=Post::model()->find($criteria); // $params tidak
diperlukan
```

Cara alternatif terhadap CDbCriteria adalah dengan mengoper array ke metode find. Kunci dan nilai array masing-masing berhubungan dengan properti kriteria nama dan nilai. Contoh di atas dapat ditulis ulang seperti berikut

```
$post=Post::model()->find(array(
    'select'=>'title',
    'condition'=>'postID=:postID',
    'params'=>array(':postID'=>10),
));
```

Ketika lebih dari satu baris data memenuhi kondisi query yang ditetapkan, kita dapat mengambilnya sekaligus menggunakan metode findAll, masing-masing memiliki pasangan metode find, seperti yang sudah kami jelaskan.

```
// cari seluruh baris yang sesuai dengan kondisi yang
ditetapkan
$posts=Post::model()->findAll($condition,$params);
// cari seluruh baris dengan primary key yang ditetapkan
$posts=Post::model()-
>findAllByPk($postIDs,$condition,$params);
// cari seluruh baris dengan nilai atribut yang ditetapkan
$posts=Post::model()-
>findAllByAttributes($attributes,$condition,$params);
// cari seluruh baris dengan pernyataan SQL yang ditetapkan
$posts=Post::model()->findAllBySql($sql,$params);
```

Jika tidak ada yang sama dengan kondisi query, findAll akan mengembalikan array kosong. Ini berbeda dengan find yang akan mengembalikan null jika tidak menemukan apapun.

Selain metode find dan findAll seperti dijelaskan di atas, metode berikut juga disediakan demi kenyamanan:

```
// ambil sejumlah baris yang sesuai dengan kondisi yang
ditetapkan
$n=Post::model()->count($condition,$params);
// ambil sejumlah baris menggunakan pernyataan SQL yang
ditetapkan
$n=Post::model()->countBySql($sql,$params);
// periksa apakah ada satu baris yang sesuai denga kondisi
yang ditetapkan
$exists=Post::model()->exists($condition,$params);
```

5. Mengupdate record

Setelah instance AR diisi dengan nilai kolom, kita dapat mengubah dan menyimpannya kembali ke tabel database.

```
$post=Post::model()->findByPk(10);
$post->title='new post title';
$post->save(); // simpan perubahan ke database
```

Seperti yang kita lihat, kita menggunakan metode save() yang sama untuk melakukan operasi insert maupun update. Jika instance AR dibuat menggunakan operator new, pemanggilan save() akan menyisipkan baris record baru ke dalam tabel database; jika turunan AR adalah hasil dari beberapa pemanggilan metode

find atau findAll, memanggil save() akan mengupdate baris yang sudah ada dalam tabel. Bahkan, kita dapat menggunakan CActiveRecord::isNewRecord untuk mengetahui apakah turunan AR baru atau tidak.

Dimungkinkan juga untuk mengupdate satu atau beberapa baris dalam sebuah tabel database tanpa memanggilnya lebih dulu. AR menyediakan metode tingkat-kelas yang nyaman untuk tujuan ini:

```
// mutakhirkan baris yang sama seperti kondisi yang
ditetapkan
Post::model()->updateAll($attributes,$condition,$params);
// mutakhirkan baris yang sama seperti kondisi dan primary
key yang ditetapkan
Post::model()-
>updateByPk($pk,$attributes,$condition,$params);
// mutakhirkan kolom counter dalam baris yang sesuai dengan
kondisi yang ditetapkan
Post::model()->updateCounters($counters,$condition,$params);
```

Dalam contoh di atas, $attributes adalah array nilai kolom yang diindeks oleh nama kolom; $counters adalah array nilai inkremental yang diindeks oleh nama kolom; sedangkan $condition dan $params seperti yang sudah dijelaskan dalam sub-bab sebelumnya.

6. Menghapus record

Kita juga bisa menghapus baris data jika turunan AR sudah diisi dengan baris ini.

```
$post=Post::model()->findByPk(10); // menganggap ada tulisan
yang memiliki ID = 10
$post->delete(); // hapus baris dari tabel database
```

Metode tingkat kelas berikut disediakan untuk menghapus baris tanpa haris mengambilnya lebih dulu:

```
// hapus baris yang sesuai dengan kondisi yang ditetapkan
Post::model()->deleteAll($condition,$params);
// hapus baris yang sesuai dengan kondisi dan primary key
yang ditetapkan
Post::model()->deleteByPk($pk,$condition,$params);
```

2.2.2 Query Builder

Query Builder adalah cara penulisan query menggunakan fungsi. Fitur ini membantu developer untuk menggunakan properti dan metode untuk menentukan bagian-bagian dari perintah SQL. Kemudian sebuah perintah SQL dieksekusi lebih lanjut oleh metode DAO seperti yang dideskripsikan pada *Data Access Objects*. *Query Builder* sangat cocok digunakan ketika memerlukan menggabungkan sebuah perintah SQL secara prosedural, atau berdasarkan suatu kondisi logis dalam aplikasi.

Query Builder sangat cocok digunakan ketika Anda memerlukan menggabungkan sebuah statement SQL secara prosedural, atau berdasarkan suatu kondisi logis dalam aplikasi Anda. Manfaat utama dalam menggunakan Query Builder termasuk:

- Memungkinkan membangun statement SQL yang kompleks secara programatik

- Fitur ini akan memberikan quote pada nama table dan kolom secara otomatis guna mencegah konflik dengan tulisan SQL ataupun karakter khusus.
- Fitur ini juga memberikan quote pada nilai parameter dan melakukan binding pada parameter ketika memungkinkan, sehingga mengurangi resiko terserang SQL injection.
- Fitur ini menyediakan sekian tingkatan abstraksi pada DB, yang menyederhanakan migrasi ke platform DB yang berbeda.

2.2.3 Data Access Object

Data Access Objects menyediakan API generik untuk mengakses data yang disimpan dalam sistem manajemen database(DBMS). Yii DAO dibangun di atas PHP Data Objects (PDO) dan menyediakan akses data gabungan untuk beberapa DBMS popular, seperti MySQL dan PostgreSQL.

Yii DAO terdiri dari empat kelas utama sebagai berikut:

- CDbConnection: mewakili koneksi ke database.
- CDbCommand: mewakili pernyataan SQL untuk dijalankan pada database.
- CDbDataReader: mewakili forward-only stream terhadap baris dari set hasil query.
- CDbTransaction: mewakili transaksi DB.

1. Membuat koneksi database

Untuk membuat koneksi database, buat instance CDbConnection dan mengaktifkannya. Nama sumber data (DSN) diperlukan untuk menetapkan

informasi yang diperlukan untuk menyambung ke database. Nama pengguna dan kata sandi juga diperlukan guna melakukan koneksi. Exception akan dimunculkan seandainya kesalahan terjadi selama pelaksanaan koneksi (misalnya DSN tidak benar atau username/password tidak benar).

```
$connection=new CDbConnection($dsn,$username,$password);
// melakukan koneksi. Anda dapat mencoba try...catch
exception yang mungkin
$connection->active=true;
......
$connection->active=false;  // tutup koneksi
```

Format DSN tergantung pada driver PDO database yang digunakan. Secara umum, DSN terdiri dari nama driver PDO, diikuti oleh titik dua, diikuti oleh sintaks koneksi spesifik-driver. Lihat Dokumentasi PDO untuk informasi lebih lengkap. Karena CDbConnection diturunkan dari CApplicationComponent, kita juga dapat menggunakannya sebagai komponen aplikasi.

2. Menjalankan pernyataan SQL

Setelah koneksi database terlaksana, pernyataan SQL dapat dijalankan menggunakan CDbCommand. Membuat instance CDbCommand dengan memanggil CDbConnection::createCommand() dengan pernyataan SQL yang ditetapkan:

```
$connection=Yii::app()->db;   // asumsi bahwa Anda memiliki
koneksi "db" yang terkonfigurasi
// Jika tidak, Anda bisa membuat sebuah koneksi secara
eksplisit
// $connection=new CDbConnection($dsn,$username,$password);
$command=$connection->createCommand($sql);
```

```
// jika diperlukan, statement SQL dapat diupdate sebagai
berikut
// $command->text=$newSQL;
```

Pernyataan SQL dijalankan via CDbCommand dalam dua cara berikut:

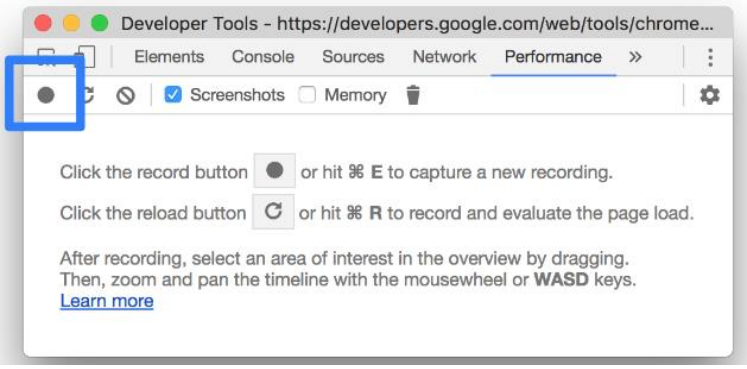
- execute(): melakukan pernyataan SQL non-query, seperti INSERT, UPDATE and DELETE. Jika berhasil, mengembalikan sejumlah baris yang dipengaruhi oleh eksekusi
- query(): melakukan pernyataan SQL yang mengembalikan baris data, seperti SELECT. Jika berhasil,mengembalikan instance CDbDataReader yang dapat ditelusuri baris data yang dihasilkan. Untuk kenyamanan, satu set metode queryXXX() juga diimplementasikan yang secara langsung mengembalikan hasil query.

Exception akan dimunculkan jika kesalahan terjadi selama eksekusi pernyataan SQL.

```
$rowCount=$command->execute();   // jalankan SQL non-query
$dataReader=$command->query();   // jalankan query SQL
$rows=$command->queryAll();      // query dan kembalikan
seluruh baris hasil
$row=$command->queryRow();       // query dan kembalikan
baris pertama hasil
$column=$command->queryColumn(); // query dan kembalikan
kolom pertama hasil
$value=$command->queryScalar();  // query dan kembalikan
field pertama dalam baris pertama
```

2.2.4 Chrome Developer Tools

Chrome developer tools merupakan tool yang disediakan oleh browser google chrome. Tools ini memberikan banyak fungsi untuk mendevelop aplikasi web. Salah satunya adalah performa analisis, fitur ini disediakan untuk membantu developer menganalisis performa dari aplikasi web yang dibuat. Tools ini bekerja dengan cara merekam runtime dari aplikasi web yang kita buat. Terdapat dua cara untuk merekam runtime dari aplikasi web. Yang pertama adalah dengan merekam semua operasi yang kita lakukan pada aplikasi web. Kemudian yang kedua adalah merekam dengan cara memuat ulang aplikasi web yang akan di analisis. Adapun user interface dari chrome developer tools ini terdapat pada gambar 2.1 Chrome Developer Tools.

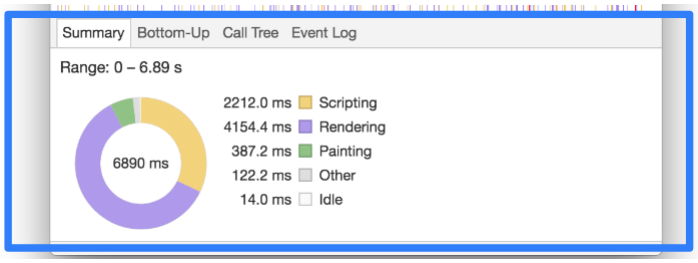


Gambar 2.1 Chrome Developer Tools

Adapun cara membuka tools ini terdapat tiga cara yaitu sebagi berikut :

1. Pilih **more tools >** **Developer Tools** dari menu chrome.
2. Klik kanan pada elemen halaman dan pilih Inspect.
3. Menggunakan shortcut pada keyboard yaitu Ctrl+Shift+I (untuk windows) atau Cmd+Opt+I (Mac).

Kemudian untuk menganalisa hasil dari pengujian, developer dapat melihat hasil dari analisis Frame per Second (FPS) yang terdapat pada hasil pengujian. Selain itu developer juga bisa melihat grafik CPU yang dihasilkan saat pengujian berlangsung. Selain itu developer juga mendapat ringkasan dari pengujian performa yang sudah selesai. Seperti pada gambar 2.2 dibawah ini.



Gambar 2.2 Hasil Pengujian