PERBANDINGAN *ISOLATION LEVEL* PADA TRANSAKSI KARTU RENCANA STUDI (KRS) (STUDI PADA KRS *ONLINE* UNIVERSITAS XYZ)

Satrio Agung W, R. Arief Setyawan, Admaja Dwi Herlambang

Abstrak: *Concurrency control* berfungsi untuk memastikan apakah *concurrent access* pengguna dilakukan secara benar. *Concurrency control* memiliki suatu kebijakan untuk menangani *lock* yang disebut dengan *isolation level*. Pemilihan *isolation level* menjadi salah satu hal yang paling penting pada transaksi Kartu Rencana Studi (KRS) *online* di Universitas XYZ. Penggunaan *isolation level* bertujuan untuk menjaga konsistensi kapasitas kelas terhadap peserta kelas dalam proses KRS *online*. Kesalahan dalam pemilihan dan implementasi *isolation level* dapat mengakibatkan proses KRS *online* menjadi lambat atau *deadlock*. Penelitian bertujuan untuk melakukan perbandingan *isolation level* dan menemukan *isolation level* yang tepat pada transaksi KRS *online*. *Isolation level* yang telah dibandingkan adalah *Uncommited Read* (UR), *Repeatable Read* (RR), *Read Stability* (RS), *Cursor Stability* (CS) dan *Cursor Stability* (CS) *with Currently Committed* (CC). Temuan penelitian menunjukkan bahwa penggunaan *isolation level* UR, RS, CS, CS *with* CC tidak mampu menjaga konsistensi kapasitas kelas. *Isolation level* RR mampu menjaga konsistensi kapasitas kelas dan rata-rata waktu eksekusi sangat cepat, 0.008850886 detik. Pengguunaan *isolation level* RR harus memperhatikan penggunaan klausa *WHERE* jika pada tabel terdapat *PRIMARY KEY*. Semua *PRIMARY KEY* harus diletakkan pada klausa *WHERE* untuk menghindari *locking* pada tingkat tabel yang bisa mengakibatkan terjadinya *deadlock*.

Kata kunci: *DBMS, concurrency control, isolation level, repeatable read, database*.

Kartu Rencana Studi (KRS) adalah kartu yang berisikan daftar mata kuliah yang akan diambil mahasiswa pada semester tertentu. Di Universitas XYZ pelaksanaan KRS dilakukan secara *online* dan secara serempak beberapa angkatan bersamaan sehingga membuat para mahasiswa harus berebut kelas sesuai dengan jadwal yang mereka inginkan. Dalam sistem KRS *online* ini basis data mempunyai peranan yang sangat penting di dalam menjaga itegritas data, salah satunya adalah menjaga konsistensi kapasitas kelas supaya tidak terisi melebihi kapasitas maksimal kelas. Untuk menangani hal tersebut maka pemilihan *isolation level* pada *concurrency control* basis data harus dilakukan secara tepat.

Concurrency control adalah proses memanajemen operasi yang dilakukan secara simultan pada sebuah database sehingga integritas data terjaga dan operasi yang dilakukan tidak diinterfensi oleh yang lain di dalam sebuah sistem *multiuser* (Hoffer, Ramesh, & Topi, 2011). Dalam melakukan *concurrency control* ada suatu kebijakan yang yang harus diambil dalam menangani *lock* yang disebut dengan *isolation level*. Pentingnya pemilihan *isolation level* di dalam *concurrency control* pada sistem basis data mirip dengan pentingnya lampu lalu lintas dalam sistem lalu lintas karena tidak adanya itu maka dapat terjadi tabrakan, kemacetan atau kebuntuan (Dimitrios Liarokapis, 2001). Pada kasus KRS *online* pentingnya pemilihan *isolation level* yang akan diimplementasikan pada *query* adalah agar dapat menjaga konsistensi kapasitas kelas namu jika

Satrio Agung adalah Dosen Fakultas Ilmu Komputer Universitas Brawijaya
R. Arief Setyawan adalah Dosen Fakultas Ilmu Komputer Universitas Brawijaya
Admaja Dwi Herlambang adalah Dosen Fakultas Ilmu Komputer Universitas Brawijaya

terjadi kesalahan dalam memilih *isolation level* maka akan terjadi *deadlock* pada sistem basis data yang menyebabkan kegagalan pada transaksi KRS *online*.

Sebuah transaksi atau *unit of work* terdiri dari satu atau lebih pernyataan *Structured Query Language* (SQL) yang mana ketika dijalankan harus diperlakukan sebagai sebuah unit, jika satu dari pernyataan di dalam transaksi gagal, maka seluruh transaksi gagal dan pernyataan yang dijalankan sampai titik kegagalan akan dilakukan *rollback*. Dalam melakukan transaksi harus memenuhi aturan ACID, yaitu *Atomic*, *Consistent*, *Isolated* dan *Durable* (Hollins, 2000). *Atomic* merupakan keadaan semua pernyataan SQL di dalam transaksi diperlakukan sebagai sebuah unit, jika semua transaksi sukses maka akan dilakukan commit, namun jika terjadi kegagalan pada transaksi maka semua transaksi akan dibatalkan atau dikembalikan ke poin tertentu. *Consistent* berarti transaksi tidak boleh melanggar constraints yang didefinisikan oleh pengguna. *Isolated* berarti transaksi yang sedang berjalan tidak boleh diinterfensi oleh transaksi yang lain. *Durable* berarti sekali sebuah transaksi commit, maka perubahan data akan bertahan di dalam basis data.

Concurrency berarti bahwa beberapa pengguna bisa bekerja pada waktu yang sama pada objek-objek basis data yang sama. *Data Base Management System* (DBMS) didesain sebagai sebuah basis data multi-user. Akses ke data harus dikoordinasikan secara tepat dan secara transparan menggunakan sebuah mekanisme untuk memastikan itegritas dan konsistensi data. *Concurrency control* dapat mencegah aplikasi lain untuk memperbarui baris sama pada suatu tabel jika baris tersebut sedang diperbarui oleh transaksi yang lain dengan cara menggunkan mekanisme *locking*. *Lock* dibutuhkan secara otomatis untuk mendukung sebuah transaksi dan akan dilepas ketika transaksi selesai (menggunakan *COMMIT* atau *ROLLBACK*). *Lock* dapat diterjadi pada baris atau tabel. Ada dua tipe dasar dari *lock*, yaitu *Share Lock*s dan *Exclusive Lock*s (Chong, Hakes, & Ahuja, 2009). *Share lock*s (*S lock*s) diperoleh ketika sebuah aplikasi ingin membaca dan mencegah yang lain untuk merubah pada baris yang sama. *Exclusive Lock*s (*X lock*s) diperoleh ketika sebuah aplikasi melakukan *update, insert*, atau *delete* sebuah baris.

Jika transaksi pada sistem basis data dieksekusi secara berurutan dengan tidak ada *overlap* ada waktu yang sama, maka transaksi tidak membutuhkan *concurrency control*. Namun, jika transaksi terjadi secara bersamaan dan saling menyisipi satu sama lain maka hasil yang tidak diinginkan dapat terjadi (Sumit Kumar, Ms. Ritu Devi,2011). Permasalahan yang dapat terjadi dengan tidak adanya *concurrency control*, yaitu *lost update*, *uncommitted read*, *non-repeatable read*, dan *phantom read* (Chong, Hakes, & Ahuja, 2009).

Isolation level dapat dianggap sebagai suatu kebijakan dalam melakukan *locking*, berdasarkan pada *isolation level* kita bisa mendapatkan perilaku yang berbeda dari *locking* basis data dalam sebuah aplikasi. Pada studi kasus penelitian ini DBMS yang digunakan adalah DB2 yang juga mendukung sejumlah *isolation level* untuk menangani *concurrency control*, yaitu *Repeatable Read* (RR), *Read Stability* (RS), *Cursor Satbility* (CS), dan *Uncommitted Read* (UR). Pada *repeadtable read* akan melakukan *lock* terhadap semua baris yang diakses selama sebuah *unit of work* (UOW). Jika sebuah aplikasi melakukan pernyataan *SELECT* dua kali di dalam UOW yang sama, maka hasil yang sama

dikembalikan. Pada RR, *lost update*s, aksess ke *uncommitted data*, *non-repeatable reads*, dan *phantom read* tidak akan terjadi.

Read stability isolation level hanya melakukan *lock* pada baris yang diambil oleh aplikasi selama UOW. RS memastikan bahwa baris yang dibawah sebuah UOW tidak bisa diubah oleh proses aplikasi yang lain sampai UOW selesai, dan perubahan yang dilakukan pada sebuah baris oleh aplikasi yang lain tidak bisa dibaca sampai berubahan dilakukan *commit*. Pada RS, akses ke *uncommitted data* dan *non-repeatable reads* tidak mungkin terjadi, namun *phantom read* masih mungkin.

Isolation level dengan *cursosr stability* akan melakukan *lock* sementara pada baris yang sedang diakses selama sebuah transaksi dimana posisi *cursor* berada. *Lock* akan terus terjadi sampai baris selanjutnya dibaca atau transaksi dihentikan. Sehingga, jika data di dalam baris telah diubah, *lock* akan menahannya sampai perubahan dilakukan commit. Pada *isolation level* ini, aplikasi lain tidak bisa *update* atau *delete* sebuah baris data dimana *cursor* berada pada. Akses *uncommitted data* oleh aplikasi lain tidak akan terjadi pada *cursor stability* namun *non-repeateble read* dan *phantom read* masih mungkin terjadi.

Uncommitted read adalah *isolation level* yang mengijinkan sebuah aplikasi untuk dapat membaca perubahan data yang belum dilakukan proses *commit* dari dari transaksi yang lain. UR tidak akan melakukan pencegahan terhadap aplikasi lain untuk mengakses baris yang sedang dibaca kecuali kalau aplikasi mencoba untuk merubah atau menghapus tabel. Pada UR, akses ke *uncommitted data*, *non-repeatable reads*, dan *phantom reads* masih mungkin terjadi. UR cocok jika kita menjalankan *query* pada *read-only* tabel atau jika kita hanya mengeluarkan pernyataan *SELECT* saja dan melihat data yang belum dilakukan *commit* oleh aplikasi yang lain.

Dari uraian yang telah dijelaskan maka peneliti akan membandingkan performansi, integritas data, serta ada tidaknya *deadlock* dari setiap *isolation level* pada sistem basis data KRS *online* yang digunakan oleh Universitas XYZ yaitu *isolation level Uncommited Read* (UR), *Repeatable Read* (RR), *Read Stability* (RS) *Cursor Stability* (CS), dan *Cursor Stability* (CS) *with Currently Committed* (CC). Sehingga hasil penelitian ini diharapkan dapat memberikan rekomendasi dalam pemilihian *isolation level* yang tepat supaya dapat dijadikan pertimbangan dan selanjutnya diimplementasikan pada *query* dimana proses KRS *online* dijalankan.

METODE

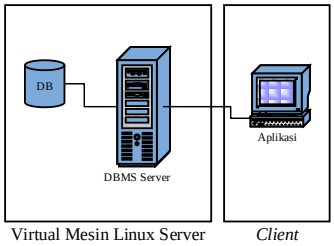
Alur metode penelitian terdiri dari kegiatan studi pustaka, analisis kebutuhan, implementasi, pengujian dan analisis, dan pengambilan kesimpulan. Studi pustaka berarti mempelajari mengenai konsep *concurrency control*, *isolation level*, serta cara mengimpelentasikannya di dalam *query* pada sistem manajemen basis data yang ada di jurnal, buku, atau paper. Analisis kebutuhan berarti melakukan analisa terhadap *query* yang sudah diimplementasikan pada transaksi KRS *online* dengan tujuan untuk mendapatkan informasi mengenai tabel apa saja yang dibutuhkan dalam proses KRS *online* serta mengetahui pola aksesnya. Implementasi berarti dari hasil analisa *query* peneliti akan melakukan impelementasi *isolation level* pada transakaksi KRS *online*. Pada tahap ini, semua *isolation level* yang didukung oleh DBMS akan diimplementasikan pada *query* yang terkait dengan transaksi

KRS *online*.yang membutuhkan penanganan *isolation level*. *Isolation level* yang akan dibandingkan yaitu UR, RR, RS, CS dan CS *with* CC.

Pada tahap pengujian, peneliti membangun sebuah perangkat lunak untuk menguji implementasi dari *isolation level* secara *concurrent access* untuk mensimulasikan sesuai dengan kondisi yang sebenarnya pada waktu KRS *online*. Pada pengujian dilakukan dengan skenario yang sama seperti pada waktu KRS *online* dimana mahasiswa secara bersamaan mengakses jadwal yang sama untuk bisa lebih dulu masuk di kelas yang dinginkan dan transaksi data yang dilakukan pada pengujian ini yaitu penambahan KRS dan pengahapusan KRS. Pada simulasi pengujian akan dibuat dua model transaksi dimana masing-masing terdapat 200 *thread* transaksi KRS yang akan memperebutkan kelas yang sama dimana transaksi pertama yang dilakukan hanya menambahkan KRS sedangkan pada transaksi kedua sebagian melakukan penghapusan KRS dan yang lain melakukan penambahan KRS. Dari hasil pengujian akan dilakukan analisis untuk dapat mengetahui perbandingan dari berbagai *isolation level* yang diterapkan mulai dari dampak terhadap kapasitas kelas, lama eksekusi serta ada tidaknya *deadlock* dalam transaksi. Kesimpulan yang diambil dalam penelitian ini adalah berdasarkan implementasi yang diterapkan, hasil pengujian, dan analisis.

HASIL

Implementasi sistem digunakan untuk membuat simulasi pengujian proses KRS *online* dengan sejumlah simultan akses dan dengan berbagai macam perbedaan *isolation level*. Arsitektur sistem simulasi pengujian proses KRS *online* dapat dilihat pada Gambar 1.



Gambar 1. Arsitektur Sistem Simulasi Pengujian Proses KRS *Online*

Pada Gambar 1 terdapat *virtual* mesin linux *server* dan *client*. Lingkungan pengujian mesin *virtual* memiliki beberapa spesifikasi, yaitu sistem operasi CentOS 6.7, 4 Core CPU, 12 GB RAM, dan DB2 ESE v10.5. Sementara pada sisi *client* di dalamnya dipasang perangkat lunak yang berfungsi untuk menjalankan proses *multithread* KRS *online* menggunakan desktop aplikasi yang dapat dilihat pada Gambar 2.

Gambar 2. DBMS Scalability Benchmark Tool

Pada sisi DBMS implementasi uji coba perbandingan *isolation level* diletakkan pada sebuah user defined function dengan nama DUMMY.GETSISAKAPASITASKELAS yang ditunjukkan pada Gambar 3.

```
CREATE OR REPLACE FUNCTION DUMMY.GETSISAKAPASITASKELAS
(  IN_TAHUN SMALLINT,   IN_IS_GANJIL      VARCHAR(2), IN_IS_PENDEK
CHARACTER(1),
   IN_KELAS VARCHAR(3), IN_K_MK            VARCHAR(15),  IN_THN_MK
            SMALLINT,
   IN_K_PROG_STUDI VARCHAR(3), IN_K_JURUSAN VARCHAR(3),
IN_K_FAKULTAS VARCHAR(3),
   IN_K_JENJANG              VARCHAR(3), IN_K_TIPE_SELEKSI
            VARCHAR(3) )
 RETURNS SMALLINT
P1: BEGIN
    DECLARE V_SISAKELAS SMALLINT;
    DECLARE V_MAX_KELAS SMALLINT;

    SELECT JADWAL_MK.MAX_PESERTA INTO V_MAX_KELAS
    FROM DB2ADMIN.JADWAL_MK JADWAL_MK
    WHERE JADWAL_MK.TAHUN = IN_TAHUN
          AND JADWAL_MK.IS_GANJIL = IN_IS_GANJIL
          AND JADWAL_MK.IS_PENDEK = IN_IS_PENDEK
          AND JADWAL_MK.KELAS = IN_KELAS
          AND JADWAL_MK.K_MK = IN_K_MK
          AND JADWAL_MK.THN_MK= IN_THN_MK
          AND JADWAL_MK.K_PROG_STUDI = IN_K_PROG_STUDI
          AND JADWAL_MK.K_JURUSAN = IN_K_JURUSAN
          AND JADWAL_MK.K_FAKULTAS = IN_K_FAKULTAS
          AND JADWAL_MK.K_JENJANG = IN_K_JENJANG
          AND JADWAL_MK.K_TIPE_SELEKSI = IN_K_TIPE_SELEKSI;

    RETURN  (SELECT COALESCE(V_MAX_KELAS-COUNT(KELAS),0)
          FROM DUMMY.MHS_KRS_KELAS_TOTAL
          WHERE TAHUN = IN_TAHUN
               AND IS_GANJIL = IN_IS_GANJIL
               AND IS_PENDEK = IN_IS_PENDEK
               AND KELAS = IN_KELAS
               AND K_MK = IN_K_MK
               AND THN_MK= IN_THN_MK
               AND K_PROG_STUDI = IN_K_PROG_STUDI
               AND K_JURUSAN = IN_K_JURUSAN
               AND K_FAKULTAS = IN_K_FAKULTAS
               AND K_JENJANG = IN_K_JENJANG
               AND K_TIPE_SELEKSI = IN_K_TIPE_SELEKSI WITH RR);
END P1
```

Gambar 3. Source Code User Defined Function Tempat Implementasi Perbandingan *Isolation level*

Pengujian *isolation level* dilakukan dengan cara merubah *source code* pada keyword WITH dengan tambahan RR, RS, CS, dan UR serta dengan menghilangkan keyword WITH untuk uji coba pada default behavior yaitu CS WITH CC. Untuk selanjutnya *user defined function* DUMMY.GETSISAKAPASITASKELAS akan diimplementasikan pada *stored procedure* yang bernama DUMMY.ADDMHSKRS yang berfungsi untuk menambahkan KRS mahasiswa, dan selanjutnya *stored procedure* tersebutlah yang akan dijalankan secara simultan untuk menguji perbandingan *isolation level*. *Source code* dari *stored procedure* DUMMY.ADDMHSKRS dapat dilihat pada Gambar 4, Gambar 5, dan Gambar 6.

```
CREATE OR REPLACE PROCEDURE DUMMY.ADDMHSKRS (
   IN IN_NIM   VARCHAR(15),
   IN IN_TAHUN                SMALLINT,
   IN IN_IS_GANJIL            VARCHAR(2),
   IN IN_IS_PENDEK            CHARACTER(1),
   IN IN_KELAS                VARCHAR(3),
   IN IN_K_MK VARCHAR(15),
   IN IN_THN_MK               SMALLINT,
   IN IN_K_PROG_STUDI         VARCHAR(3),
   IN IN_K_JURUSAN            VARCHAR(3),
   IN IN_K_FAKULTAS           VARCHAR(3),
   IN IN_K_JENJANG            VARCHAR(3),
   IN IN_K_TIPE_SELEKSI       VARCHAR(3),
   IN IN_USERID               VARCHAR(45),
   IN IN_ANGKATAN_KTM         SMALLINT,
   IN IN_K_FAKULTAS_KTM       VARCHAR(3),
   IN IN_K_JENJANG_KTM        VARCHAR(3),
   IN IN_K_JURUSAN_KTM        VARCHAR(3),
   IN IN_K_PROG_STUDI_KTM VARCHAR(3) )
 SPECIFIC SQL150121143650883
 DYNAMIC RESULT SETS 1
P1: BEGIN
  DECLARE VAR_START_TIME TIMESTAMP;
  DECLARE SQLSTATE     CHAR(5) DEFAULT '00000';
  DECLARE VAR_ERRSTATE CHAR(5);
  DECLARE VAR_MESSAGE VARCHAR(150);
  DECLARE VAR_SEMESTER INT;
  DECLARE VAR_ANGKATAN INT;
  DECLARE VAR_MIN_SKS INT;
  DECLARE VAR_K_PROG_STUDI                VARCHAR(3);
  DECLARE VAR_K_JURUSAN  VARCHAR(3);
  DECLARE VAR_K_FAKULTAS VARCHAR(3);
  DECLARE VAR_K_JENJANG  VARCHAR(3);
  DECLARE VAR_TOTAL_MKSYARAT SMALLINT;
  DECLARE VAR_IP_BEBAN DOUBLE;
  DECLARE VAR_IPK_BEBAN DOUBLE;
  DECLARE VAR_SKSK_BEBAN DOUBLE;
  DECLARE VAR_STATUS_CEK_SYARAT CHAR(1);

  DECLARE RS1 CURSOR WITH RETURN FOR
    SELECT VAR_ERRSTATE AS ERRSTATE,
         VAR_MESSAGE AS MESSAGE,
         DEC(CURRENT TIMESTAMP-VAR_START_TIME,9,6) AS ELAPSED_TIME
    FROM SYSIBM.SYSDUMMY1;
  DECLARE CONTINUE   HANDLER FOR NOT FOUND
    SET VAR_ERRSTATE = SQLSTATE;
  DECLARE CONTINUE HANDLER FOR SQLWARNING
    SET VAR_ERRSTATE = SQLSTATE;
  DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    SET VAR_ERRSTATE = SQLSTATE;

  SET VAR_START_TIME=CURRENT TIMESTAMP;
  SET VAR_ERRSTATE=SQLSTATE;


  set VAR_ANGKATAN=IN_ANGKATAN_KTM;
  set VAR_K_FAKULTAS=IN_K_FAKULTAS_KTM;
  set VAR_K_JENJANG=IN_K_JENJANG_KTM;
  set VAR_K_JURUSAN=IN_K_JURUSAN_KTM;
  SET VAR_K_PROG_STUDI=IN_K_PROG_STUDI_KTM;

  --info akademik
  SELECT  A.IPK_BEBAN, A.IP_BEBAN, A.SKSK_BEBAN INTO
VAR_IPK_BEBAN,VAR_IP_BEBAN,VAR_SKSK_BEBAN
  FROM DB2ADMIN.MHS_AKADEMIK A
  WHERE  A.NIM = IN_NIM AND
       A.SEMESTER = AKADE-
MIK.GETMHSSEMESTER(VAR_ANGKATAN,IN_TAHUN,IN_IS_GANJIL) - 1 AND
       A.IS_PENDEK = IN_IS_PENDEK;

  --JIKA PADA SEMESTER CURRENT -1 TIDAK DITEMUKAN NILAI AKADEMIK MAKA
DIAMBIL DARI SEMESTER PALING AKHIR
  IF VAR_ERRSTATE = '02000' THEN
--      IF (VAR_IPK_BEBAN IS NULL) OR (VAR_IP_BEBAN IS NULL) OR
(VAR_SKSK_BEBAN IS NULL) THEN
          SET VAR_ERRSTATE = '00000';
          --info akademik
          SELECT  A.IPK_BEBAN, A.IP_BEBAN, A.SKSK_BEBAN INTO
VAR_IPK_BEBAN,VAR_IP_BEBAN,VAR_SKSK_BEBAN
          FROM DB2ADMIN.MHS_AKADEMIK A
          WHERE  A.NIM = IN_NIM AND A.IS_PENDEK = IN_IS_PENDEK
          ORDER BY A.SEMESTER DESC
          FETCH FIRST ROWS ONLY;
--      END IF;
  END IF;

    --cek kalender masa krs
  IF DUM-
MY.CEKKALENDERKRS(VAR_ANGKATAN,IN_TAHUN,IN_IS_GANJIL,IN_IS_PENDEK,

VAR_K_PROG_STUDI,VAR_K_JURUSAN,VAR_K_FAKULTAS,VAR_K_JENJANG) = '1'
```

Gambar 4. Source Code Stored Procedure untuk Menambah KRS Mahasiswa

```
THEN
    A:
    IF DUMMY.CEKKELASBENTROK(IN_NIM, IN_TAHUN,IN_IS_GANJIL,
IN_IS_PENDEK,IN_KELAS,
                        IN_K_MK, IN_THN_MK,
IN_K_PROG_STUDI,IN_K_JURUSAN,IN_K_FAKULTAS,
                        IN_K_JENJANG,IN_K_TIPE_SELEKSI) = '0' THEN
      IF DUMMY.CEKKRSBENTROK( IN_NIM, IN_TAHUN,IN_IS_GANJIL,
IN_IS_PENDEK,IN_KELAS,
                        IN_K_MK, IN_THN_MK,
IN_K_PROG_STUDI,IN_K_JURUSAN,IN_K_FAKULTAS,
                        IN_K_JENJANG,IN_K_TIPE_SELEKSI) = '0' THEN
--CEK SISA KREDIT
        IF (DUM-
MY.GETSISAKREDITSKS(IN_NIM,VAR_K_PROG_STUDI,VAR_K_JURUSAN,V
AR_K_FAKULTAS,VAR_K_JENJANG,

IN_K_MK,IN_THN_MK,IN_IS_GANJIL,IN_IS_PENDEK,VAR_IP_BEBAN,VAR_
IPK_BEBAN) >= 0)   THEN
          SET VAR_STATUS_CEK_SYARAT = DUM-
MY.CEKMKSYARAT(IN_NIM,IN_K_MK,IN_THN_MK,VAR_K_PROG_STUDI,

VAR_K_JURUSAN,VAR_K_FAKULTAS,VAR_K_JENJANG,
                                        VAR_SKSK_BEBAN);
          IF VAR_STATUS_CEK_SYARAT = '1' THEN
            SET  VAR_ERRSTATE = '00004'; --MINIMUM SKS TIDAK
TERPENUHI
          ELSEIF VAR_STATUS_CEK_SYARAT = '2' THEN
            SET  VAR_ERRSTATE = '00005'; --MK PRASYARAT BELUM
DITEMPUH
          END IF;
        ELSE
          SET VAR_ERRSTATE = '00003'; --MAKSIMAL KREDIT TIDAK
MEMENUHI
        END IF;
      ELSE
        SET VAR_ERRSTATE = '00002'; --JADWAL BENTROK
      END IF;
    ELSE
      SET VAR_ERRSTATE = '00007'; --KELAS DOUBLE
    END IF;
  ELSE
    SET VAR_ERRSTATE = '00001'; -- JADWAL SUDAH DITUTUP/BELUM
DIBUKA
  END IF;

  IF (VAR_ERRSTATE = '00000') THEN
    IF (DUMMY.GETSISAKAPASITASKELAS(IN_TAHUN, IN_IS_GANJIL,
IN_IS_PENDEK,
                        IN_KELAS, IN_K_MK, IN_THN_MK,
                        IN_K_PROG_STUDI, IN_K_JURUSAN,
IN_K_FAKULTAS,
                        IN_K_JENJANG, IN_K_TIPE_SELEKSI) > 0)
THEN
      INSERT INTO DUMMY.MHS_KRS_KELAS(NIM, TAHUN,
IS_GANJIL, IS_PENDEK, KELAS, K_MK, THN_MK,
                        K_PROG_STUDI, K_JURUSAN,
K_FAKULTAS, K_JENJANG, K_TIPE_SELEKSI,
                        K_SALAH, IS_SETUJU, IS_HAPUS,
IS_TAMBAH, USERID)
      VALUES (IN_NIM, IN_TAHUN, IN_IS_GANJIL, IN_IS_PENDEK,
IN_KELAS, IN_K_MK, IN_THN_MK,
          IN_K_PROG_STUDI, IN_K_JURUSAN, IN_K_FAKULTAS,
IN_K_JENJANG, IN_K_TIPE_SELEKSI,
          '0','0', '0', '0', IN_USERID);
      INSERT INTO DUMMY.MHS_KELAS(NIM, TAHUN, IS_GANJIL,
IS_PENDEK, KELAS, K_MK, THN_MK,
                        K_PROG_STUDI, K_JURUSAN, K_FAKULTAS,
K_JENJANG, K_TIPE_SELEKSI,
                        K_TIPE_NILAI, NILAI_KE, NILAI, USERID)
      VALUES (IN_NIM, IN_TAHUN, IN_IS_GANJIL, IN_IS_PENDEK,
IN_KELAS, IN_K_MK, IN_THN_MK,
          IN_K_PROG_STUDI, IN_K_JURUSAN, IN_K_FAKULTAS,
IN_K_JENJANG, IN_K_TIPE_SELEKSI,
          '1' , 1, 0, IN_USERID);

      INSERT INTO DUMMY.MHS_KHS(NIM, SEMESTER, IS_PENDEK,
K_MK, THN_MK, K_NILAI, IS_HAPUS, IS_TAMBAH, USERID)
      VALUES (IN_NIM, AKADE-
MIK.GETMHSSEMESTER(VAR_ANGKATAN,IN_TAHUN,IN_IS_GANJIL),
IN_IS_PENDEK, IN_K_MK, IN_THN_MK,
          'K', '0', '0', IN_USERID);
    ELSE
      SET VAR_ERRSTATE = '00006'; --KAPASITAS KELAS PENUH
    END IF;
  END IF;

  IF VAR_ERRSTATE = '00000' THEN
    COMMIT;
    SET VAR_MESSAGE='KRS berhasil ditambahkan';
  ELSE
    ROLLBACK;
    IF VAR_ERRSTATE = '00001' THEN
      SET VAR_MESSAGE='Jadwal KRS ditutup';
    ELSEIF VAR_ERRSTATE = '00002' THEN
      SET VAR_MESSAGE='Jadwal kuliah bersinggungan dengan jadwal
yang ada di KRS';
    ELSEIF VAR_ERRSTATE = '00003' THEN
      SET VAR_MESSAGE='Maksimal kredit anda tidak mencukupi';
    ELSEIF VAR_ERRSTATE = '00004' THEN
      SET VAR_MESSAGE='Minimum sks untuk mengambil MK belum
terpenuhi';
```

Gambar 5. Lanjutan Source Code Stored Procedure untuk Menambah KRS Mahasiswa

```
    ELSEIF VAR_ERRSTATE = '00005' THEN
      SET VAR_MESSAGE='Mata Kuliah prasyarat belum terpenuhi';
    ELSEIF VAR_ERRSTATE = '00006' THEN
      SET VAR_MESSAGE='Jumlah peserta kelas sudah penuh';
    ELSEIF VAR_ERRSTATE = '00007' THEN
      SET VAR_MESSAGE='Mata kuliah sudah diambil di kelas lain';
    ELSEIF VAR_ERRSTATE = '40001' THEN
      SET VAR_MESSAGE='Proses KRS timeout dikarenakan antrian
pada kelas yg sama terlalu banyak, KRS gagal ditambahkan, silahkan
mencoba lagi';
    ELSE
      SET VAR_MESSAGE='KRS gagal ditambahkan';
    END IF;
  END IF;

  OPEN RS1;
END P1
```

Gambar 6. Lanjutan Source Code Stored Procedure untuk Menambah KRS Mahasiswa

PEMBAHASAN

Hasil pengujian dari implentasi *isolation level* pada proses menambah KRS dapat dilihat pada Tabel 1 dan Tabel 2.

Tabel 1. Hasil Pengujian Terhadap Konsistensi maksimal Kapasitas Kelas dan Waktu Eksekusi Pada Saat Menambah KRS

(1)	(2)		(3)		(4)
	A	B	A	B	
CS WITH	4	4	4	4	0.00590452
CC	5	5	5	6	3
UR	4	4	4	4	0.015675
	5	5	5	5	
CS	4	4	4	4	0.01236
	5	5	5	5	
RS	4	4	4	4	0.014405
	5	5	7	5	
RR	4	4	4	4	0.01236
	5	5	5	5	

Keterangan:
(1) *Isolation level*
(2) Kapasitas kelas
(3) Jumlah KRS masuk
(4) Rata-rata waktu eksekusi (detik)

Pada Tabel 1 dapat dilihat bahwa konsistensi terhadap kapasitas maksimal pada saat proses penambahan KRS hanya dapat dijaga oleh *isolation level* RR, UR dan CS dengan rata-rata waktu eksekusi yang relatif sama. Namun ketika pada transaksi KRS ditambahkan dengan adanya proses penghapusan KRS maka hanya *Isolation level* RR mampu menjaga konsistensi kapasitas kelas.

Tabel 2. Hasil Pengujian Terhadap Konsistensi maksimal Kapasitas Kelas dan Waktu Eksekusi Pada Saat Menambah dan Penghapusan KRS

(1)	(2)		(3)		(4)
	A	B	A	B	
CS WITH	4	4	4	4	0.00811666
CC	5	5	5	4	7
UR	4	4	4	4	0.00881512
	5	5	7	6	6
CS	4	4	4	4	0.00557322
	5	5	5	4	2
RS	4	4	4	4	0.00589873
	5	5	4	5	4
RR	4	4	4	4	0.00534177
	5	5	5	5	2

Keterangan:
(1) *Isolation level*
(2) Kapasitas kelas
(3) Jumlah KRS masuk
(4) Rata-rata waktu eksekusi (detik)

Pada Tabel 2 dapat dilihat bahwa jumlah maksimal kelas pada *Isolation level* RR sama dengan jumlah KRS yang masuk dan tidak ada kelas yang jumlah pesertanya kurang dari kapasitas maksimal kelas. Dengan demikian maka *isolation level* yang sesuai dengan proses KRS ada repeatable read (RR) karena mampu menjaga konsistensi kapasitas maksimal kelas dan jika dibuat rata-rata waktu eksekusi pada Tabel 1 dan Tabel 2 juga sangat cepat yaitu 0.008850886 detik.

SIMPULAN & SARAN

Dari penelitian yang sudah dilakukan maka dapat ditarik beberapa kesimpulan. Pertama, *isolation level* pada KRS *online* diimplementasikan di *query* yang ada di dalam user defined function pada waktu menghitung jumlah peserta kelas. Kedua, pengujian *isolation level* dilakukan dengan membangun sebuah perangkat lunak yang mampu melakukan thread yang secara simultan melakukan proses KRS. Ketiga, *isolation level* UR, RS, CS, CS with CC masih memungkinkan terjadinya kelebihan atau kurangnya kapasitas kelas sementara *Isolation level* RR mampu menjaga konsistensi maksimal kapasitas kelas. Keempat, pada penggunaan *isolation level* RR harus diperhatikan penggunaan klausa WHERE jika pada tabel terdapat PRIMARY KEY, seluruh PRIMARY KEY harus diletakkan pada klausa WHERE untuk menghindari *locking* pada tingkat tabel yang bisa mengakibatkan terjadinya *deadlock*.

Pengujian pada penelitian ini masih berfokus pada ketahanan dari DBMS belum menggunakan mekanisme yang sama seperti pada KRS *online*. Untuk penelitian selanjutnya, pengujian dapat ditambahkan dengan menggunakan web service yang secara simultan mengakses ke DBMS sehingga dapat diketahui dampaknya terhadap kecepatan akses web service.

DAFTAR RUJUKAN

Chong, R, Hakes, I., & Ahuja, R. 2009. *Getting Started with DB2 Express-C: 3th Edition*. USA: IBM Corporation.

Hoffer, J.A., Ramesh, V., & Topi, H. 2011. *Modern Database Management System: Tenth Edition*. New York: Prentice Hall.

Hollins, M. 2000. *Transaction Isolation levels and Object Oriented Data Structures*, (Online), (https://cs.adelaide.edu.au/~idea/idea7/PDFs/hollins.pdf), diakses 5 Januari 2015).

Kumar, S., & Devi, R. 2011. An Analytical Review of Orientation Based Concurrency. *Global Journal of Computer Science and Technology*. 11(23): hlm. 22-26.