

# E-GENTING PROGRAMMING COMPETITION, 2004

## WORKSHOP HANDOUT, WEEK 2, VERSION 1

### 1 BACKGROUND

#### 1.1 B-Tree Operations

```
template <typename Key_t, typename Row_t> class BTree_c {
public:
    void BtReset () { /*...*/ }
        // Position the row pointer at the beginning of
        // the tree
    bool BtFind (const Key_t *key) { /*...*/ }
        // Position the row pointer at the row with key
        // 'key', or if no such row exists, the row with
        // the next highest key. Return 1 if the key
        // was found, otherwise 0.
    bool BtRead (Row_t *row) { /*...*/ }
        // Read the row that the row pointer is pointing
        // to and load it into the location addressed by
        // 'row'. Move the row pointer to the next row
        // in the tree. Return 1 if a row was read.
        // Return 0 if the row pointer is pointing to the
        // end of the tree.
};
```

#### 1.2 An Example Schema

```
// Customer file
create table custFile (
    custId    integer not null,    // Customer identifier
    custName  char(40) not null   // Customer name
);
create index custIdInd on custFile (custId);

// Purchases file
create table purFile (
    purId     integer not null,    // Customer identifier
    purNo     integer not null,    // Purchase number
    purVal    integer not null     // Purchase value
);
create index purIdNoInd on purFile (purId, purNo);
```

### 1.3 Query Translation

Select statement	<pre>select custName, purVal from   custFile, purFile where  custId &gt;= 100127 and        purId = custId;</pre>
Access strategy	<pre>cfKey.custId = 100127; custFile.BtFind (&amp;cfKey); while (custFile.BtRead(&amp;cfRec)) {     pfKey.purId = cfRec.custId;     pfKey.purNo = MIN_PUR_NO;     purFile.BtFind(&amp;pfKey);     while (         purFile.BtRead(&amp;pfRec) &amp;&amp;         pfRec.purId == cfRec.custId     )         emit (cfRec.custName, pfRec.purVal); }</pre>

### 1.4 An Alternative Translation

Select statement	<pre>select custName, purVal from   custFile, purFile where  custId &gt;= 100127 and        purId = custId;</pre>
Access strategy	<pre>cfKey.custId = 100127; custFile.BtFind (&amp;cfKey); pfKey.purId = 100127; pfKey.purNo = MIN_PUR_NO; purFile.BtFind (&amp;pfKey); pfOK = purFile.BtRead (&amp;pfRec); while (custFile.BtRead(&amp;cfRec)) {     while (pfOK &amp;&amp; pfRec.purId &lt;= cfRec.custId) {         if (pfRec.purId == cfRec.custId)             emit (cfRec.custName, pfRec.purVal);         pfOK = purFile.BtRead (&amp;pfRec);     } }</pre>

## 2 EXERCISES

### 2.1 The Schema for the Exercises

```
create table execs (  
    execCode integer not null,  
    execName char(20) not null,  
);  
create index execCodeInd on execs(execCode);  
  
create table loans (  
    loanId integer not null,  
    loanCust char(20) not null,  
    loanAmount double precision not null,  
    loanStatus smallint not null  
);  
create index loanIdInd on loans(loanId);  
  
create table approvals (  
    appLoanId integer not null,  
    appExecCode integer not null,  
    appType smallint not null  
);  
create index appLoanIdInd on  
    approvals (appLoanId, appExecCode);
```

### 2.2 Execution Times and Database Parameters

Reset operation	0.3ms
Find operation	0.7ms
Read operation	0.8ms
Number of rows in the execs table	1,000
Number of rows in the loans table	1,000,000
Number of rows in the approvals table	typically 2 per loan

## 2.3 Problems

Determine a b-tree access strategy for each of the following queries and estimate the execution time of the query.

1	<pre>select  execCode, execName from    execs where   execCode between 100000 and 100999;</pre>
2	<pre>select  sum(loanAmount) from    loans where   loanCust = 'JOE BLOGS' and         loanStatus = 0;</pre>
3	<pre>select  loanId, appType, loanAmount from    loans, approvals where   appExecCode = 100240 and         loanId = appLoanId;</pre>
4	<pre>select  count(*), sum(loanAmount) from    loans, approvals where   loanId &gt;= 872 and         appLoanId = loanId;</pre>
5	<pre>select  execCode, execName, count(*), sum(loanAmount) from    approvals, loans, execs where   loanId = appLoanId and         appExecCode = execCode and         loanStatus = 0 group by execCode, execName;</pre>