

Interfacing with MySQL

A GUI to
browse a
MySQL table

- 1 A GUI to browse a MySQL table
our database with Python scripts
connecting to database, setting sort order
retrieving and displaying records

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

- 2 Normalization
splitting a table in two
moving data into the tables

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

- 3 Retrieving Specific Records
use the key to access a record
inserting a new record in a table

use the key to
access a record
inserting a new
record in a table

MCS 275 Lecture 26
Programming Tools and File Management
Jan Verschelde, 12 March 2010

Interfacing with MySQL

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

1 A GUI to browse a MySQL table our database with Python scripts

connecting to database, setting sort order
retrieving and displaying records

2 Normalization

splitting a table in two
moving data into the tables

3 Retrieving Specific Records

use the key to access a record
inserting a new record in a table

our Database of Python Scripts

viewing it with a GUI

We create one table `scripts` in the database
`OurPyFiles`, storing data about our Python scripts.

For every Python script we have 4 fields:
its type, number, date, and file name.

Example of a header:

```
# L-26 MCS 275 Fri 12 Mar 2010 : guidb1.py
```

Corresponding data tuple:

```
('L', '26', '2010-03-12', 'guidb1.py')
```

With a GUI we get a better overview.
We will sort the records in various ways.

our Database of Python Scripts

viewing it with a GUI

We create one table `scripts` in the database `OurPyFiles`, storing data about our Python scripts.

For every Python script we have 4 fields:
its type, number, date, and file name.

Example of a header:

```
# L-26 MCS 275 Fri 12 Mar 2010 : guidb1.py
```

Corresponding data tuple:

```
('L', '26', '2010-03-12', 'guidb1.py')
```

With a GUI we get a better overview.

We will sort the records in various ways.

our Database of Python Scripts

viewing it with a GUI

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record

inserting a new
record in a table

We create one table `scripts` in the database
`OurPyFiles`, storing data about our Python scripts.

For every Python script we have 4 fields:
its type, number, date, and file name.

Example of a header:

```
# L-26 MCS 275 Fri 12 Mar 2010 : guidb1.py
```

Corresponding data tuple:

```
('L', '26', '2010-03-12', 'guidb1.py')
```

With a GUI we get a better overview.

We will sort the records in various ways.

our Database of Python Scripts

viewing it with a GUI

We create one table `scripts` in the database `OurPyFiles`, storing data about our Python scripts.

For every Python script we have 4 fields:
its type, number, date, and file name.

Example of a header:

```
# L-26 MCS 275 Fri 12 Mar 2010 : guidb1.py
```

Corresponding data tuple:

```
('L', '26', '2010-03-12', 'guidb1.py')
```

With a GUI we get a better overview.
We will sort the records in various ways.

Widgets in the GUI

Message field to write status information.

Buttons to . . .

- connect to the database
- retrieve all records
- see next 10 or previous 10 records

Radio buttons to determine

- which field to use for sorting
- sort in ascending or descending order

Widgets in the GUI

Message field to write status information.

Buttons to . . .

- connect to the database
- retrieve all records
- see next 10 or previous 10 records

Radio buttons to determine

- which field to use for sorting
- sort in ascending or descending order

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

Widgets in the GUI

Message field to write status information.

Buttons to . . .

- connect to the database
- retrieve all records
- see next 10 or previous 10 records

Radio buttons to determine

- which field to use for sorting
- sort in ascending or descending order

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

Widgets in the GUI

Message field to write status information.

Buttons to . . .

- connect to the database
- retrieve all records
- see next 10 or previous 10 records

Radio buttons to determine

- which field to use for sorting
- sort in ascending or descending order

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

Widgets in the GUI

Message field to write status information.

Buttons to . . .

- connect to the database
- retrieve all records
- see next 10 or previous 10 records

Radio buttons to determine

- which field to use for sorting
- sort in ascending or descending order

Widgets in the GUI

Message field to write status information.

Buttons to . . .

- connect to the database
- retrieve all records
- see next 10 or previous 10 records

Radio buttons to determine

- which field to use for sorting
- sort in ascending or descending order

12 Mar 2010

Viewing the Table scripts

A GUI to browse a MySQL table

our database with Python scripts

connecting to database, setting sort order

retrieving and displaying records

Normalization

splitting a table in two
moving data into the tables

Retrieving Specific Records

use the key to access a record
inserting a new record in a table

GUI to OurPyFiles db

retrieved 77 records

connect retrieve next 10 previous 10

type	date	file
L-12	2008-02-11	findsecret.py
L-12	2008-02-11	binarysearch.py
Q-5	2008-02-12	permutations.py
L-13	2008-02-13	selectsort.py
L-13	2008-02-13	quicksort.py
L-13	2008-02-13	mergesort.py
L-14	2008-02-15	facstack.py
L-14	2008-02-15	gcdstack.py
L-14	2008-02-15	fibstack.py
L-14	2008-02-15	evalpostfix.py

sort by type date file

ascending order descending order

Interfacing with MySQL

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

1 A GUI to browse a MySQL table

our database with Python scripts

connecting to database, setting sort order

retrieving and displaying records

2 Normalization

splitting a table in two

moving data into the tables

3 Retrieving Specific Records

use the key to access a record

inserting a new record in a table

Message Label and other labels

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

Code in `__init__`:

```
self.message = StringVar()
self.message.set("welcome to our database")
self.messageLabel = Label(wdw, \
    textvariable = self.message)
self.messageLabel.grid(row=0, column=0, columnspan=4)

self.tt = Label(wdw, text='type')
self.tt.grid(row=2, column=0)
self.dd = Label(wdw, text='date')
self.dd.grid(row=2, column=1)
self.ff = Label(wdw, text='file')
self.ff.grid(row=2, column=2, columnspan=2)

self.sL = Label(wdw, text='sort by')
self.sL.grid(row=4, column=0)
```

Message Label and other labels

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

Code in `__init__`:

```
self.message = StringVar()
self.message.set("welcome to our database")
self.messageLabel = Label(wdw, \
    textvariable = self.message)
self.messageLabel.grid(row=0, column=0, columnspan=4)

self.tt = Label(wdw, text='type')
self.tt.grid(row=2, column=0)
self.dd = Label(wdw, text='date')
self.dd.grid(row=2, column=1)
self.ff = Label(wdw, text='file')
self.ff.grid(row=2, column=2, columnspan=2)

self.sL = Label(wdw, text='sort by')
self.sL.grid(row=4, column=0)
```

Message Label and other labels

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

Code in `__init__`:

```
self.message = StringVar()
self.message.set("welcome to our database")
self.messageLabel = Label(wdw, \
    textvariable = self.message)
self.messageLabel.grid(row=0, column=0, columnspan=4)

self.tt = Label(wdw, text='type')
self.tt.grid(row=2, column=0)
self.dd = Label(wdw, text='date')
self.dd.grid(row=2, column=1)
self.ff = Label(wdw, text='file')
self.ff.grid(row=2, column=2, columnspan=2)

self.sL = Label(wdw, text='sort by')
self.sL.grid(row=4, column=0)
```

Connecting to the Database

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record

inserting a new
record in a table

Button defined in `__init__`:

```
self.bc = Button(wdw, text='connect', \
                 command = self.connect)
self.bc.grid(row=1,column=0)
self.cursor = 0
```

```
def connect(self):
```

```
    """
```

```
    Connects to the database OurPyFiles.
```

```
    """
```

```
    try:
```

```
        db = MySQLdb.connect(db="OurPyFiles")
```

```
        self.message.set("connected to \"OurPyFiles\"")
```

```
        self.cursor = db.cursor()
```

```
    except:
```

```
        self.message.set("failed to connect to \"OurPyFiles\"")
```

Connecting to the Database

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record

inserting a new
record in a table

Button defined in `__init__`:

```
self.bc = Button(wdw, text='connect', \  
                 command = self.connect)  
self.bc.grid(row=1,column=0)  
self.cursor = 0
```

```
def connect(self):
```

```
    """
```

```
    Connects to the database OurPyFiles.
```

```
    """
```

```
    try:
```

```
        db = MySQLdb.connect(db="OurPyFiles")
```

```
        self.message.set("connected to \"OurPyFiles\"")
```

```
        self.cursor = db.cursor()
```

```
    except:
```

```
        self.message.set("failed to connect to \"OurPyFiles\"")
```

Radio Button to sort

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

We can sort on type, date, or file.

Code for RadioButton in `__init__`:

```
self.RadioSort = IntVar()
self.st = Radiobutton(wdw, text='type', \
    variable = self.RadioSort, value = 1)
self.st.grid(row=4,column=1)
self.sd = Radiobutton(wdw, text='date', \
    variable = self.RadioSort, value = 2)
self.sd.grid(row=4,column=2)
self.sf = Radiobutton(wdw, text='file', \
    variable = self.RadioSort, value = 3)
self.sf.grid(row=4,column=3)
self.RadioSort.set(1)
```

The default sort is set to type.

Radio Button to sort

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

We can sort on type, date, or file.

Code for RadioButton in `__init__`:

```
self.RadioSort = IntVar()
self.st = Radiobutton(wdw, text='type', \
    variable = self.RadioSort, value = 1)
self.st.grid(row=4,column=1)
self.sd = Radiobutton(wdw, text='date', \
    variable = self.RadioSort, value = 2)
self.sd.grid(row=4,column=2)
self.sf = Radiobutton(wdw, text='file', \
    variable = self.RadioSort, value = 3)
self.sf.grid(row=4,column=3)
self.RadioSort.set(1)
```

The default sort is set to type.

Radio Button to sort

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

We can sort on type, date, or file.

Code for RadioButton in `__init__`:

```
self.RadioSort = IntVar()
self.st = Radiobutton(wdw, text='type', \
    variable = self.RadioSort, value = 1)
self.st.grid(row=4,column=1)
self.sd = Radiobutton(wdw, text='date', \
    variable = self.RadioSort, value = 2)
self.sd.grid(row=4,column=2)
self.sf = Radiobutton(wdw, text='file', \
    variable = self.RadioSort, value = 3)
self.sf.grid(row=4,column=3)
self.RadioSort.set(1)
```

The default sort is set to type.

Radio Button to sort

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

We can sort on type, date, or file.

Code for RadioButton in `__init__`:

```
self.RadioSort = IntVar()
self.st = Radiobutton(wdw, text='type', \
    variable = self.RadioSort, value = 1)
self.st.grid(row=4,column=1)
self.sd = Radiobutton(wdw, text='date', \
    variable = self.RadioSort, value = 2)
self.sd.grid(row=4,column=2)
self.sf = Radiobutton(wdw, text='file', \
    variable = self.RadioSort, value = 3)
self.sf.grid(row=4,column=3)
self.RadioSort.set(1)
```

The default sort is set to type.

Radio Button to sort

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

We can sort on type, date, or file.

Code for RadioButton in `__init__`:

```
self.RadioSort = IntVar()
self.st = Radiobutton(wdw, text='type', \
    variable = self.RadioSort, value = 1)
self.st.grid(row=4,column=1)
self.sd = Radiobutton(wdw, text='date', \
    variable = self.RadioSort, value = 2)
self.sd.grid(row=4,column=2)
self.sf = Radiobutton(wdw, text='file', \
    variable = self.RadioSort, value = 3)
self.sf.grid(row=4,column=3)
self.RadioSort.set(1)
```

The default sort is set to type.

Radio Button to sort

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

We can sort on type, date, or file.

Code for RadioButton in `__init__`:

```
self.RadioSort = IntVar()
self.st = Radiobutton(wdw, text='type', \
    variable = self.RadioSort, value = 1)
self.st.grid(row=4,column=1)
self.sd = Radiobutton(wdw, text='date', \
    variable = self.RadioSort, value = 2)
self.sd.grid(row=4,column=2)
self.sf = Radiobutton(wdw, text='file', \
    variable = self.RadioSort, value = 3)
self.sf.grid(row=4,column=3)
self.RadioSort.set(1)
```

The default sort is set to type.

Radio Button to order

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

User has choice between ascending or descending order.

Code for RadioButton in `__init__`:

```
self.RadioOrder = IntVar()
self.asc = Radiobutton(wdw, text='ascending order',
    variable = self.RadioOrder, value = 1)
self.asc.grid(row=5, column=0, columnspan=2)
self.dsc = Radiobutton(wdw, text='descending order',
    variable = self.RadioOrder, value = 2)
self.dsc.grid(row=5, column=2, columnspan=2)
self.RadioOrder.set(1)
```

The default order is set to ascending.

Radio Button to order

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

User has choice between ascending or descending order.

Code for RadioButton in `__init__`:

```
self.RadioOrder = IntVar()  
self.asc = Radiobutton(wdw, text='ascending order',  
                        variable = self.RadioOrder, value = 1)  
self.asc.grid(row=5, column=0, columnspan=2)  
self.dsc = Radiobutton(wdw, text='descending order',  
                        variable = self.RadioOrder, value = 2)  
self.dsc.grid(row=5, column=2, columnspan=2)  
self.RadioOrder.set(1)
```

The default order is set to ascending.

Radio Button to order

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

User has choice between ascending or descending order.

Code for RadioButton in `__init__`:

```
self.RadioOrder = IntVar()  
self.asc = Radiobutton(wdw, text='ascending order',  
                        variable = self.RadioOrder, value = 1)  
self.asc.grid(row=5, column=0, columnspan=2)  
self.dsc = Radiobutton(wdw, text='descending order',  
                        variable = self.RadioOrder, value = 2)  
self.dsc.grid(row=5, column=2, columnspan=2)  
self.RadioOrder.set(1)
```

The default order is set to ascending.

Radio Button to order

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

User has choice between ascending or descending order.

Code for RadioButton in `__init__`:

```
self.RadioOrder = IntVar()  
self.asc = Radiobutton(wdw, text='ascending order',  
                        variable = self.RadioOrder, value = 1)  
self.asc.grid(row=5, column=0, columnspan=2)  
self.dsc = Radiobutton(wdw, text='descending order',  
                        variable = self.RadioOrder, value = 2)  
self.dsc.grid(row=5, column=2, columnspan=2)  
self.RadioOrder.set(1)
```

The default order is set to ascending.

Radio Button to order

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

User has choice between ascending or descending order.

Code for RadioButton in `__init__`:

```
self.RadioOrder = IntVar()  
self.asc = Radiobutton(wdw, text='ascending order',  
                        variable = self.RadioOrder, value = 1)  
self.asc.grid(row=5, column=0, columnspan=2)  
self.dsc = Radiobutton(wdw, text='descending order',  
                        variable = self.RadioOrder, value = 2)  
self.dsc.grid(row=5, column=2, columnspan=2)  
self.RadioOrder.set(1)
```

The default order is set to ascending.

Formulating the Query

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def query(self):
    """
    Returns the query to select all records,
    taking input from the radio buttons.
    """
    q = 'select * from scripts'
    ord = ''
    if self.RadioOrder.get() == 1:
        ord = 'asc'
    elif self.RadioOrder.get() == 2:
        ord = 'desc'
    if self.RadioSort.get() == 1:
        q = q + ' order by t, n ' + ord
    elif self.RadioSort.get() == 2:
        q = q + ' order by d ' + ord
    elif self.RadioSort.get() == 2:
        q = q + ' order by f ' + ord
    return q
```

Formulating the Query

A GUI to browse a MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def query(self):
    """
    Returns the query to select all records,
    taking input from the radio buttons.
    """
    q = 'select * from scripts'
    ord = ''
    if self.RadioOrder.get() == 1:
        ord = 'asc'
    elif self.RadioOrder.get() == 2:
        ord = 'desc'
    if self.RadioSort.get() == 1:
        q = q + ' order by t, n ' + ord
    elif self.RadioSort.get() == 2:
        q = q + ' order by d ' + ord
    elif self.RadioSort.get() == 2:
        q = q + ' order by f ' + ord
    return q
```

Formulating the Query

A GUI to browse a MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def query(self):
    """
    Returns the query to select all records,
    taking input from the radio buttons.
    """
    q = 'select * from scripts'
    ord = ''
    if self.RadioOrder.get() == 1:
        ord = 'asc'
    elif self.RadioOrder.get() == 2:
        ord = 'desc'
    if self.RadioSort.get() == 1:
        q = q + ' order by t, n ' + ord
    elif self.RadioSort.get() == 2:
        q = q + ' order by d ' + ord
    elif self.RadioSort.get() == 2:
        q = q + ' order by f ' + ord
    return q
```

Interfacing with MySQL

A GUI to
browse a
MySQL table

1 A GUI to browse a MySQL table

our database with Python scripts
connecting to database, setting
sort order
retrieving and
displaying records

Normalization

2 Normalization

splitting a table in two
moving data into the tables

Retrieving
Specific
Records

3 Retrieving Specific Records

use the key to access a record
inserting a new record in a table

splitting a table in two
moving data into the
tables

use the key to
access a record
inserting a new
record in a table

Retrieving Records

second button

```
def retrieve(self):
    """
    Retrieves all records from the scripts table.
    """
    if self.cursor == 0:
        self.message.set("please connect first")
    else:
        q = self.query()
        lc = self.cursor.execute(q)
        m = 'retrieved %d records' % int(lc)
        self.message.set(m)
        R = self.cursor.fetchall()
        self.clear()
        for i in range(0, len(R)):
            if i >= 10: break
            self.insert(R[i])
        self.records = R
        self.pos = 10
```

Retrieving Records

second button

```
def retrieve(self):
    """
    Retrieves all records from the scripts table.
    """
    if self.cursor == 0:
        self.message.set("please connect first")
    else:
        q = self.query()
        lc = self.cursor.execute(q)
        m = 'retrieved %d records' % int(lc)
        self.message.set(m)
        R = self.cursor.fetchall()
        self.clear()
        for i in range(0, len(R)):
            if i >= 10: break
            self.insert(R[i])
        self.records = R
        self.pos = 10
```

Retrieving Records

second button

```
def retrieve(self):
    """
    Retrieves all records from the scripts table.
    """
    if self.cursor == 0:
        self.message.set("please connect first")
    else:
        q = self.query()
        lc = self.cursor.execute(q)
        m = 'retrieved %d records' % int(lc)
        self.message.set(m)
        R = self.cursor.fetchall()
        self.clear()
        for i in range(0,len(R)):
            if i >= 10: break
            self.insert(R[i])
        self.records = R
        self.pos = 10
```

Retrieving Records

second button

```
def retrieve(self):
    """
    Retrieves all records from the scripts table.
    """
    if self.cursor == 0:
        self.message.set("please connect first")
    else:
        q = self.query()
        lc = self.cursor.execute(q)
        m = 'retrieved %d records' % int(lc)
        self.message.set(m)
        R = self.cursor.fetchall()
        self.clear()
        for i in range(0, len(R)):
            if i >= 10: break
            self.insert(R[i])
        self.records = R
        self.pos = 10
```

Displaying Data in Listboxes

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

To display the data, we use **Listbox** widgets.

We will use three listboxes (defined in `__init__`):

① for type and number, e.g.: L-23

```
self.Lt = Listbox(wdw,width=4,height=10)
self.Lt.grid(row=3,column=0)
```

② for the date, e.g.: 2010-03-12

```
self.Ld = Listbox(wdw,width=10,height=10)
self.Ld.grid(row=3,column=1)
```

③ for the file name, e.g.: guidb1.py

```
self.Ln = Listbox(wdw,width=15,height=10)
self.Ln.grid(row=3,column=2,columnspan=2)
```

Displaying Data in Listboxes

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

To display the data, we use **Listbox** widgets.

We will use three listboxes (defined in `__init__`):

1 for type and number, e.g.: L-23

```
self.Lt = Listbox(wdw,width=4,height=10)
self.Lt.grid(row=3,column=0)
```

2 for the date, e.g.: 2010-03-12

```
self.Ld = Listbox(wdw,width=10,height=10)
self.Ld.grid(row=3,column=1)
```

3 for the file name, e.g.: guidb1.py

```
self.Ln = Listbox(wdw,width=15,height=10)
self.Ln.grid(row=3,column=2,columnspan=2)
```

Displaying Data in Listboxes

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

To display the data, we use **Listbox** widgets.

We will use three listboxes (defined in `__init__`):

① for type and number, e.g.: L-23

```
self.Lt = Listbox(wdw,width=4,height=10)
self.Lt.grid(row=3,column=0)
```

② for the date, e.g.: 2010-03-12

```
self.Ld = Listbox(wdw,width=10,height=10)
self.Ld.grid(row=3,column=1)
```

③ for the file name, e.g.: guidb1.py

```
self.Ln = Listbox(wdw,width=15,height=10)
self.Ln.grid(row=3,column=2,columnspan=2)
```

Displaying Data in Listboxes

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

To display the data, we use **Listbox** widgets.

We will use three listboxes (defined in `__init__`):

① for type and number, e.g.: L-23

```
self.Lt = Listbox(wdw,width=4,height=10)
self.Lt.grid(row=3,column=0)
```

② for the date, e.g.: 2010-03-12

```
self.Ld = Listbox(wdw,width=10,height=10)
self.Ld.grid(row=3,column=1)
```

③ for the file name, e.g.: guidb1.py

```
self.Ln = Listbox(wdw,width=15,height=10)
self.Ln.grid(row=3,column=2,columnspan=2)
```

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

Using listboxes

```
def clear(self):
    """
    Clears all listboxes.
    """
    self.Lt.delete(0,END)
    self.Ld.delete(0,END)
    self.Ln.delete(0,END)

def insert(self,item):
    """
    Inserts one record item to the listboxes.
    """
    t = item[0] + '-' + str(int(item[1]))
    self.Lt.insert(END,t)
    self.Ld.insert(END,str(item[2]))
    self.Ln.insert(END,item[3])
```

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

Using listboxes

```
def clear(self):
    """
    Clears all listboxes.
    """
    self.Lt.delete(0,END)
    self.Ld.delete(0,END)
    self.Ln.delete(0,END)

def insert(self,item):
    """
    Inserts one record item to the listboxes.
    """
    t = item[0] + '-' + str(int(item[1]))
    self.Lt.insert(END,t)
    self.Ld.insert(END,str(item[2]))
    self.Ln.insert(END,item[3])
```

Buttons to Navigate

Because our listboxes are only 10 in height,
we cannot display everything all at once.

Two navigation buttons in `__init__`:

- 1 to show the next 10 records:

```
self.bn = Button(wdw, text='next 10', \
                 command = self.next10)
self.bn.grid(row=1, column=2)
```

- 2 to show the previous 10 records:

```
self.bp = Button(wdw, text='previous 10', \
                 command = self.prev10)
self.bp.grid(row=1, column=3)
```

Storing the current position in the retrieved records:

```
self.pos = 0
```

Buttons to Navigate

Because our listboxes are only 10 in height, we cannot display everything all at once.

Two navigation buttons in `__init__`:

1 to show the next 10 records:

```
self.bn = Button(wdw, text='next 10', \
                 command = self.next10)
self.bn.grid(row=1, column=2)
```

2 to show the previous 10 records:

```
self.bp = Button(wdw, text='previous 10', \
                 command = self.prev10)
self.bp.grid(row=1, column=3)
```

Storing the current position in the retrieved records:

```
self.pos = 0
```

Displaying next 10 Records

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

**retrieving and
displaying records**

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def next10(self):
    """
    Displays the next 10 records in listboxes.
    """
    if self.records == 0:
        self.message.set("no records to show")
    else:
        self.clear()
        for i in range(self.pos, self.pos+10):
            if i >= len(self.records): break
            self.insert(self.records[i])
        self.pos = self.pos + 10
        if self.pos >= len(self.records):
            self.pos = len(self.records) - 1
```

Displaying next 10 Records

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def next10(self):
    """
    Displays the next 10 records in listboxes.
    """
    if self.records == 0:
        self.message.set("no records to show")
    else:
        self.clear()
        for i in range(self.pos, self.pos+10):
            if i >= len(self.records): break
            self.insert(self.records[i])
        self.pos = self.pos + 10
        if self.pos >= len(self.records):
            self.pos = len(self.records) - 1
```

Displaying next 10 Records

A GUI to browse a MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def next10(self):
    """
    Displays the next 10 records in listboxes.
    """
    if self.records == 0:
        self.message.set("no records to show")
    else:
        self.clear()
        for i in range(self.pos, self.pos+10):
            if i >= len(self.records): break
            self.insert(self.records[i])
        self.pos = self.pos + 10
        if self.pos >= len(self.records):
            self.pos = len(self.records) - 1
```

Displaying previous 10 Records

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def prev10(self):
    """
    Displays the previous 10 records in listboxes.
    """
    if self.records == 0:
        self.message.set("no records to show")
    else:
        self.clear()
        self.pos = self.pos - 20
        if self.pos < 0: self.pos = 0
        for i in range(self.pos, self.pos+10):
            if i >= len(self.records): break
            self.insert(self.records[i])
        self.pos = self.pos + 10
```

Displaying previous 10 Records

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def prev10(self):
    """
    Displays the previous 10 records in listboxes.
    """
    if self.records == 0:
        self.message.set("no records to show")
    else:
        self.clear()
        self.pos = self.pos - 20
        if self.pos < 0: self.pos = 0
        for i in range(self.pos, self.pos+10):
            if i >= len(self.records): break
            self.insert(self.records[i])
        self.pos = self.pos + 10
```

Displaying previous 10 Records

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def prev10(self):
    """
    Displays the previous 10 records in listboxes.
    """
    if self.records == 0:
        self.message.set("no records to show")
    else:
        self.clear()
        self.pos = self.pos - 20
        if self.pos < 0: self.pos = 0
        for i in range(self.pos, self.pos+10):
            if i >= len(self.records): break
            self.insert(self.records[i])
        self.pos = self.pos + 10
```

Interfacing with MySQL

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

1 A GUI to browse a MySQL table
our database with Python scripts
connecting to database, setting sort order
retrieving and displaying records

2 Normalization
splitting a table in two
moving data into the tables

3 Retrieving Specific Records
use the key to access a record
inserting a new record in a table

Making the Database more useful

splitting the table scripts

The four fields were copied from the headers.

Two drawbacks:

- 1 only the file name is unique for every record and not so convenient to use as key
- 2 as most lectures featured more than one script, there is a lot of redundant data in the table

We normalize, using scripts to create two new tables: one with file names and one with type and dates.

Because tedious to do manually with the mysql monitor, we develop a script.

Making the Database more useful

splitting the table scripts

The four fields were copied from the headers.

Two drawbacks:

- 1 only the file name is unique for every record and not so convenient to use as key
- 2 as most lectures featured more than one script, there is a lot of redundant data in the table

We normalize, using scripts to create two new tables: one with file names and one with type and dates.

Because tedious to do manually with the mysql monitor, we develop a script.

Making the Database more useful

splitting the table scripts

The four fields were copied from the headers.

Two drawbacks:

- 1 only the file name is unique for every record and not so convenient to use as key
- 2 as most lectures featured more than one script, there is a lot of redundant data in the table

We normalize, using scripts to create two new tables: one with file names and one with type and dates.

Because tedious to do manually with the mysql monitor, we develop a script.

Making the Database more useful

splitting the table scripts

The four fields were copied from the headers.

Two drawbacks:

- 1 only the file name is unique for every record and not so convenient to use as key
- 2 as most lectures featured more than one script, there is a lot of redundant data in the table

We normalize, using scripts to create two new tables: one with file names and one with type and dates.

Because tedious to do manually with the mysql monitor, we develop a script.

Making the Database more useful

splitting the table scripts

The four fields were copied from the headers.

Two drawbacks:

- 1 only the file name is unique for every record and not so convenient to use as key
- 2 as most lectures featured more than one script, there is a lot of redundant data in the table

We normalize, using scripts to create two new tables: one with file names and one with type and dates.

Because tedious to do manually with the mysql monitor, we develop a script.

The Function main()

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def main():  
    """  
    Splits scripts in two tables.  
    """  
    db = MySQLdb.connect(db="OurPyFiles")  
    c = db.cursor()  
    q = 'select * from scripts order by d'  
    lc = c.execute(q)  
    print 'found %d records' % int(lc)  
    A = c.fetchall()  
    CreateTables(c)  
    (T,F) = SplitRecords(A)  
    InsertTypeDate(c,T)  
    InsertFileData(c,F)
```

The Function main()

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def main():  
    """  
    Splits scripts in two tables.  
    """  
    db = MySQLdb.connect(db="OurPyFiles")  
    c = db.cursor()  
    q = 'select * from scripts order by d'  
    lc = c.execute(q)  
    print 'found %d records' % int(lc)  
    A = c.fetchall()  
    CreateTables(c)  
    (T,F) = SplitRecords(A)  
    InsertTypeDate(c,T)  
    InsertFileData(c,F)
```

The Function main()

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def main():  
    """  
    Splits scripts in two tables.  
    """  
    db = MySQLdb.connect(db="OurPyFiles")  
    c = db.cursor()  
    q = 'select * from scripts order by d'  
    lc = c.execute(q)  
    print 'found %d records' % int(lc)  
    A = c.fetchall()  
    CreateTables(c)  
    (T,F) = SplitRecords(A)  
    InsertTypeDate(c,T)  
    InsertFileData(c,F)
```

The Function main()

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def main():  
    """  
    Splits scripts in two tables.  
    """  
    db = MySQLdb.connect(db="OurPyFiles")  
    c = db.cursor()  
    q = 'select * from scripts order by d'  
    lc = c.execute(q)  
    print 'found %d records' % int(lc)  
    A = c.fetchall()  
    CreateTables(c)  
    (T,F) = SplitRecords(A)  
    InsertTypeDate(c,T)  
    InsertFileData(c,F)
```

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

Creating the Tables

```
def CreateTable(c):  
    """  
    Executes the MySQL commands to create  
    the tables typedate and filedata.  
    The input parameter c is the cursor.  
    """  
    CreateTypeDate(c)  
    CreateFileData(c)
```

Creating the Tables

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def CreateTable(c):  
    """  
    Executes the MySQL commands to create  
    the tables typedate and filedata.  
    The input parameter c is the cursor.  
    """  
    CreateTypeDate(c)  
    CreateFileData(c)
```

Create Table typedate

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def CreateTypeDate(c):
```

```
    """
```

The table typedate stores type and date,
e.g.: L-4 and 2008-01-23, represented by
four fields: id, type, number, and date.
The input parameter c is the cursor.

```
    """
```

```
    try:
```

```
        td = 'create table typedate ' + \  
            '( i INT, t CHAR(1), n INT, d DATE )'
```

```
        c.execute(td)
```

```
        print td + ' succeeded'
```

```
    except:
```

```
        print td + ' went wrong...'
```

Create Table typedate

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def CreateTypeDate(c):
    """
    The table typedate stores type and date,
    e.g.: L-4 and 2008-01-23, represented by
    four fields: id, type, number, and date.
    The input parameter c is the cursor.
    """
    try:
        td = 'create table typedate ' + \
            '( i INT, t CHAR(1), n INT, d DATE )'
        c.execute(td)
        print td + ' succeeded'
    except:
        print td + ' went wrong...'
```

Create Table typedate

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def CreateTypeDate(c):
    """
    The table typedate stores type and date,
    e.g.: L-4 and 2008-01-23, represented by
    four fields: id, type, number, and date.
    The input parameter c is the cursor.
    """
    try:
        td = 'create table typedate ' + \
            '( i INT, t CHAR(1), n INT, d DATE )'
        c.execute(td)
        print td + ' succeeded'
    except:
        print td + ' went wrong...'
```

Create Table filedata

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def CreateFileData(c):
    """
    The table filedata stores file names,
    represented by three fields: id, name,
    and id of the corresponding entry in
    the typedate table.  c is the cursor.
    """
    try:
        fd = 'create table filedata ' + \
            '( i INT, f CHAR(20), t INT )'
        c.execute(fd)
        print fd + ' succeeded'
    except:
        print fd + ' went wrong...'
```

Create Table filedata

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def CreateFileData(c):
    """
    The table filedata stores file names,
    represented by three fields: id, name,
    and id of the corresponding entry in
    the typedate table.  c is the cursor.
    """
    try:
        fd = 'create table filedata ' + \
            '( i INT, f CHAR(20), t INT )'
        c.execute(fd)
        print fd + ' succeeded'
    except:
        print fd + ' went wrong...'
```

Create Table filedata

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def CreateFileData(c):
    """
    The table filedata stores file names,
    represented by three fields: id, name,
    and id of the corresponding entry in
    the typedate table.  c is the cursor.
    """
    try:
        fd = 'create table filedata ' + \
            '( i INT, f CHAR(20), t INT )'
        c.execute(fd)
        print fd + ' succeeded'
    except:
        print fd + ' went wrong...'
```

Interfacing with MySQL

A GUI to browse a MySQL table

our database with Python scripts
connecting to database, setting sort order
retrieving and displaying records

Normalization

splitting a table in two
moving data into the tables

Retrieving Specific Records

use the key to access a record
inserting a new record in a table

- 1 A GUI to browse a MySQL table
our database with Python scripts
connecting to database, setting sort order
retrieving and displaying records
- 2 Normalization
splitting a table in two
moving data into the tables
- 3 Retrieving Specific Records
use the key to access a record
inserting a new record in a table

Splitting Records

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def SplitRecords(R):
    """
    Returns two lists: records that go
    in typedate and in filedata.
    """
    L = []; T = []; F = []; cnt = -1
    for i in range(0,len(R)):
        d = R[i]
        p = (d[0],d[1],d[2])
        if p in L:
            ind = L.index(p)
        else:
            cnt = cnt + 1
            L.append((d[0],d[1],d[2]))
            T.append((cnt,d[0],d[1],d[2]))
            ind = cnt
        F.append((i,d[3],ind))
    return (T,F)
```

Splitting Records

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def SplitRecords(R):
    """
    Returns two lists: records that go
    in typedate and in filedata.
    """
    L = []; T = []; F = []; cnt = -1
    for i in range(0,len(R)):
        d = R[i]
        p = (d[0],d[1],d[2])
        if p in L:
            ind = L.index(p)
        else:
            cnt = cnt + 1
            L.append((d[0],d[1],d[2]))
            T.append((cnt,d[0],d[1],d[2]))
            ind = cnt
        F.append((i,d[3],ind))
    return (T,F)
```

Splitting Records

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def SplitRecords(R):
    """
    Returns two lists: records that go
    in typedate and in filedata.
    """
    L = []; T = []; F = []; cnt = -1
    for i in range(0,len(R)):
        d = R[i]
        p = (d[0],d[1],d[2])
        if p in L:
            ind = L.index(p)
        else:
            cnt = cnt + 1
            L.append((d[0],d[1],d[2]))
            T.append((cnt,d[0],d[1],d[2]))
            ind = cnt
        F.append((i,d[3],ind))
    return (T,F)
```

Splitting Records

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def SplitRecords(R):
    """
    Returns two lists: records that go
    in typedate and in filedata.
    """
    L = []; T = []; F = []; cnt = -1
    for i in range(0,len(R)):
        d = R[i]
        p = (d[0],d[1],d[2])
        if p in L:
            ind = L.index(p)
        else:
            cnt = cnt + 1
            L.append((d[0],d[1],d[2]))
            T.append((cnt,d[0],d[1],d[2]))
            ind = cnt
        F.append((i,d[3],ind))
    return (T,F)
```

Inserting Type and Date

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def InsertTypeDate(c,T):
```

```
    """
```

Given the cursor and a list of values for
the table typedate, all records are added.

```
    """
```

```
    for i in range(0,len(T)):
```

```
        p = T[i]
```

```
        q = 'insert into typedate values ' + \
            '(\ "%s\ ", \ "%s\ ", \ "%s\ ", \ "%s\ ") ' + \
            '% (str(p[0]), str(p[1]), str(p[2]), \
                str(p[3]))
```

```
        c.execute(q)
```

Inserting Type and Date

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def InsertTypeDate(c,T):
    """
    Given the cursor and a list of values for
    the table typedate, all records are added.
    """
    for i in range(0,len(T)):
        p = T[i]
        q = 'insert into typedate values ' + \
            '(\ "%s\ ", \ "%s\ ", \ "%s\ ", \ "%s\ ") ' + \
            '% (str(p[0]), str(p[1]), str(p[2]), \
                str(p[3]))
        c.execute(q)
```

Inserting File Names

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def InsertFileData(c,F):
    """
    Given the cursor and a list of values for
    the table filedata, all records are added.
    """
    for i in range(0,len(F)):
        n = F[i]
        q = 'insert into filedata values ' + \
            '("%s", "%s", "%s") ' \
            '% (str(n[0]), n[1], str(n[2]))
        c.execute(q)
```

Inserting File Names

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def InsertFileData(c,F):
    """
    Given the cursor and a list of values for
    the table filedata, all records are added.
    """
    for i in range(0,len(F)):
        n = F[i]
        q = 'insert into filedata values ' + \
            '(\ "%s\ ", \ "%s\ ", \ "%s\ ") ' \
            '% (str(n[0]), n[1], str(n[2]))'
        c.execute(q)
```

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

A Sanity Check

```
def DateFiles(c):
```

```
    """
```

```
    As a final check, selects file name  
    and corresponding date from the newly  
    created tables.  c is the cursor.
```

```
    """
```

```
    q = 'select f,d from filedata, typedate' \  
        + ' where filedata.t = typedate.i'  
    c.execute(q)  
    R = c.fetchall()  
    print R  
    print '#records :', len(R)
```

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

A Sanity Check

```
def DateFiles(c):  
    """  
    As a final check, selects file name  
    and corresponding date from the newly  
    created tables.  c is the cursor.  
    """  
    q = 'select f,d from filedata, typedate' \  
        + ' where filedata.t = typedate.i'  
    c.execute(q)  
    R = c.fetchall()  
    print R  
    print '#records :', len(R)
```

Interfacing with MySQL

A GUI to browse a MySQL table

our database with Python scripts
connecting to database, setting sort order
retrieving and displaying records

Normalization

splitting a table in two
moving data into the tables

Retrieving Specific Records

use the key to access a record
inserting a new record in a table

- 1 A GUI to browse a MySQL table
our database with Python scripts
connecting to database, setting sort order
retrieving and displaying records
- 2 Normalization
splitting a table in two
moving data into the tables
- 3 Retrieving Specific Records
use the key to access a record
inserting a new record in a table

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

Using Keys

After the normalization, every script has a unique key.
This key is in the `i` field of the table `filedata`.

Accessing the data via the key allows us to

- 1 select records directly
- 2 insert, delete, and update records.

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

Using Keys

After the normalization, every script has a unique key.
This key is in the `i` field of the table `filedata`.

Accessing the data via the key allows us to

- 1 select records directly
- 2 insert, delete, and update records.

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

Using Keys

After the normalization, every script has a unique key.
This key is in the `i` field of the table `filedata`.

Accessing the data via the key allows us to

- 1 select records directly
- 2 insert, delete, and update records.

12 Mar 2010

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

The Widgets in the GUI



Query for Specific Record

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def query(self, key):
```

```
    """
```

Returns the query for all the information
of the script with the given key.

```
    """
```

```
    q = 'select typedate.t, n, d, f ' + \
        'from typedate, filedata ' + \
        'where filedata.t = typedate.i ' + \
        'and filedata.i = %d' % key
    return q
```

Query for Specific Record

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

```
def query(self, key):  
    """  
    Returns the query for all the information  
    of the script with the given key.  
    """  
    q = 'select typedate.t, n, d, f ' + \  
        'from typedate, filedata ' + \  
        'where filedata.t = typedate.i ' + \  
        'and filedata.i = %d' % key  
    return q
```

Interfacing with MySQL

A GUI to browse a MySQL table

our database with Python scripts
connecting to database, setting sort order
retrieving and displaying records

Normalization

splitting a table in two
moving data into the tables

Retrieving Specific Records

use the key to access a record
inserting a new record in a table

- 1 A GUI to browse a MySQL table
our database with Python scripts
connecting to database, setting sort order
retrieving and displaying records
- 2 Normalization
splitting a table in two
moving data into the tables
- 3 Retrieving Specific Records
use the key to access a record
inserting a new record in a table

Counting the Number of Files

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def CountFiles(self):  
    """  
    Returns the number of files.  
    """  
  
    q = 'select count(*) from filedata'  
    r = self.cursor.execute(q)  
    n = self.cursor.fetchone()  
    return int(n[0])
```

Counting the Number of Files

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
**inserting a new
record in a table**

```
def CountFiles(self):  
    """  
    Returns the number of files.  
    """  
    q = 'select count(*) from filedata'  
    r = self.cursor.execute(q)  
    n = self.cursor.fetchone()  
    return int(n[0])
```

12 Mar 2010

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record

inserting a new
record in a table

Inserting a new File

Entering the data:



Inserting a new File

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record

inserting a new
record in a table

Asking to confirm the data:



A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

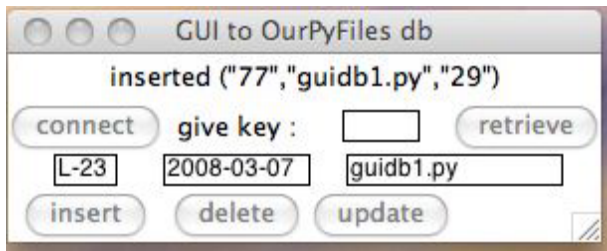
Retrieving Specific Records

use the key to
access a record

inserting a new
record in a table

Inserting a new File

Confirming the insert:



A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record

inserting a new
record in a table

Inserting Data

```
def InsertData(self):
    """
    Inserts a new record into the database.
    """
    t = self.tp.get()
    d = self.dt.get()
    f = self.nm.get()
    r = (t,d,f)
    if not self.startinsert:
        m = 'inserting %s,' % str(r)
        m = m + 'press to confirm'
        self.message.set(m)
        self.startinsert = True
    else:
        self.InsertRecord(r)
        m = 'inserted %s' % str(r)
        self.startinsert = False
```

Inserting Data

```
def InsertData(self):  
    """  
    Inserts a new record into the database.  
    """  
    t = self.tp.get()  
    d = self.dt.get()  
    f = self.nm.get()  
    r = (t,d,f)  
    if not self.startinsert:  
        m = 'inserting %s,' % str(r)  
        m = m + 'press to confirm'  
        self.message.set(m)  
        self.startinsert = True  
    else:  
        self.InsertRecord(r)  
        m = 'inserted %s' % str(r)  
        self.startinsert = False
```

Inserting Data

A GUI to
browse a
MySQL table

our database with
Python scripts

connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record

inserting a new
record in a table

```
def InsertData(self):
    """
    Inserts a new record into the database.
    """
    t = self.tp.get()
    d = self.dt.get()
    f = self.nm.get()
    r = (t,d,f)
    if not self.startinsert:
        m = 'inserting %s,' % str(r)
        m = m + 'press to confirm'
        self.message.set(m)
        self.startinsert = True
    else:
        self.InsertRecord(r)
        m = 'inserted %s' % str(r)
        self.startinsert = False
```

Inserting a Record

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record
inserting a new
record in a table

```
def InsertRecord(self,data):
    """
    Prepares data to insert a record.
    """
    L = data[0].split('-')
    t = L[0]; n = L[1]
    d = data[1]; f = data[2]
    nf = self.CountFiles()
    nt = self.CountTypes()
    v = '(' + '\"' + str(nf) + '\"' + \
        '\',\' + '\"' + f + '\"' + \
        '\',\' + '\"' + str(nt) + '\"' + \
        '\',\' + \
    self.InsertFileValues(v)
    m = 'inserted %s' % v
    self.message.set(m)
```

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record

inserting a new
record in a table

Inserting Values

Recall the MySQL command:

```
insert into <table> values <tuple>
```

```
def InsertFileValues(self,v):  
    """  
    Inserts values in filedata table.  
    """  
    q = 'insert into filedata values '  
    q = q + v  
    self.cursor.execute(q)
```

Inserting Values

A GUI to
browse a
MySQL table

our database with
Python scripts
connecting to
database, setting
sort order

retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving
Specific
Records

use the key to
access a record

inserting a new
record in a table

Recall the MySQL command:

```
insert into <table> values <tuple>
```

```
def InsertFileValues(self,v):  
    """  
    Inserts values in filedata table.  
    """  
    q = 'insert into filedata values '  
    q = q + v  
    self.cursor.execute(q)
```

Summary and Assignments

A GUI to browse a MySQL table

our database with
Python scripts
connecting to
database, setting
sort order
retrieving and
displaying records

Normalization

splitting a table in two
moving data into the
tables

Retrieving Specific Records

use the key to
access a record
inserting a new
record in a table

A good reference on GUIs with Tkinter:

<http://infohost.nmt.edu/tcc/help/pubs/tkinter.pdf>

visit <http://www.mysqltutorial.org/>

Exercises:

- 1 Create the equivalent to `guidb1.py`, our first GUI, writing a CGI script.
- 2 The method in our second GUI to insert is not yet complete. Also provide functions to add the corresponding values in the table `typedate`.
- 3 Provide functions to delete records.
- 4 Provide functions to update records.