

## Databasestøttet Webpublicering, forår 2002

### Forelæsning 10

#### Online Communities

- Online communities
- Online community features
- ACS - Arsdigita Community System
- Gennemgang af 7 vigtige moduler
- Oracle sekvenser
- Mere om check af formvariable
- A changable site-design
- Introduktion til Øvelse 9
  - group\_by
  - Generering af stjerner med rekursion

#### Online community features

Et online community skal:

1. indeholde data om brugere
2. indeholde information om statiske sider knyttet til foreningen
  - Hvem bidrog med siden?
  - Er siden tilknyttet andre sider?
3. holde styr på hvilke brugere har set hvilke sider
4. holde styr på hvilke brugere som koster foreningen penge og tid
5. holde styr på hvorledes brugere kommer til foreningen og hvilke eksterne links brugere følger
6. holde styr på banner-ads - og brugerens brug af dem
7. hjælpe administrator med at kontakte grupper af brugere

#### Eksempler på brug:

- Joe har et spørgsmål: Undersøg om Joe har læst sider og fulgt diskussioner om emnet (kræver modul 1 og 3)
- Eksperten: Undersøg om der er huller i de statiske sider der kan udfyldes (kræver modul 2, 3 og 4)
- Brugtkøb: En køber skal kunne se sælgers historie i foreningen (kræver modul 1)

#### Online communities

Online communities (foreninger) har mange fordele:

- Vi kan lære uden at være tilknyttet et universitet
- Vi kan undervise uden at undervise fuldtid
- Vi kan lære uden at sige vores job op
- Vi kan lære uden at forlade vores hjem
- Vi kan samarbejde uden at være i samme bygning
- Vi kan bidrage til projekter uden at arbejde fuldtid
- Vi kan møde ligesindede

#### ACS - Arsdigita Community System

En samling moduler til opbygning af web-sites med mulighed for bruger-interaktion.

Syv moduler:

1. Brugerdatabase
2. Information om indhold
3. Information om hvad en bruger har læst
4. Værdien af en bruger
5. Hvilke links har brugere fulgt for at komme til sitet
6. Banner-ads
7. Håndtering af tilknytninger mellem administrator og brugere

## Modul 1: Brugerdatabase

Hierfelt ACS:

```
create table users (
  user_id          integer not null primary key,
  first_names     varchar(100) not null,
  last_name       varchar(100) not null,
  priv_name       integer default 0,
  email           varchar(100) not null unique,
  priv_email      integer default 5,
  email_bouncing_p char(1) default 'F',
                  check(email_bouncing_p in ('T','F')),
  password        varchar(30) not null,
  url             varchar(200),
  on_vacation_until date,
  last_visit      date,
  second_to_last_visit date,
  registration_date date
);
```

## Modul 2: Information om indhold

Tabeller til at håndtere:

- Statistiske sider
  - Brugerinndastede kommentarer
  - Brugerinndastede relaterede links
  - Classified ads :  
Camera dealer:  
<http://www.photo.net/wtr/thebook/screenshots/community-member.camera-beale>
- Fotograf:  
<http://www.photo.net/wtr/thebook/screenshots/community-member.reader.html>
- Chat
  - Diskussionsgrupper

## Nyt og gammelt indhold

Hvordan adskilles

“Nyt indhold siden sidste login”

fra

“Indhold der allerede er set” ?

Når bruger logger på, så sættes

last\_visit = sysdate.

second\_to\_last\_visit = last\_visit

hvis last\_visit enten er mere end en dag gammel eller slet ikke findes.

Når vi skal præsentere nyheder, så præsentierer vi alt som er *nyere* end second-to-last-visit.

## Modul 3: Information om hvad en bruger har læst

Brugbart når en bruger stiller spørgsmål:

- Har brugeren læst det relevante materiale?

## Modul 4: Værdien af en bruger

- Hvor tit skal en administrator sætte en brugers kommentar eller spørgsmål?
- Giver bruger gode svar på spørgsmål stillet af andre brugere?
- Opgiver en bruger en ikke-virkende email-adresse?
- Stiller en bruger gode spørgsmål?
- Hvor tit skal en administrator sætte en kommentar fra en bruger?

### Modul 5: Hvilke links har brugere fulgt for at komme til sitet?

— og hvilke lokale sider bliver brugere sendt til fra de eksterne sites?

```
create table referer_log (  
  -- relative to the PageRoot, includes the leading /  
  local_url varchar(250) not null,  
  
  -- full URL on the foreign server (including http://)  
  foreign_url varchar(250) not null,  
  -- we count referrals per day  
  
  entry_date date,  
  click_count integer default 0,  
  primary key ( local_url, foreign_url, entry_date)  
);
```

Eksempel:

```
local_url /photo/  
foreign_url http://www.yahoo.com/Arts/....  
entry_date 1998-06-12  
click_count 24
```

### Modul 7: Håndtering af tilknytninger mellem administrator og brugere

Følgende kriterier skal kunne bruges til at kontakte grupper af brugere:

- Aktivitet i diskussionsgrupper
- Har læst en special statisk side
- Har kommenteret en special statisk side
- Har skrevet en statisk side
- Medlemsid af foreningen
- Har udtrykt interesse for et emne

### Modul 6: Banner ads

Her er datamodellen for håndtering af banner ads:

```
create table advs (  
  adv_key          varchar(200) primary key,  
  -- picture  
  adv_filename     varchar(200),  
  target_url       varchar(500)  
);  
  
create table adv_log (  
  adv_key          varchar(200) not null references advs,  
  entry_date      date,  
  display_count   integer default 0,  
  click_count     integer default 0,  
  unique(adv_key, entry_date)  
);
```

### OpenACS - ACS baseret på PostgreSQL

- OpenACS (<http://www.openacs.org>) er baseret på open source software:
  - AOLserver: <http://www.aolserver.com>
  - PostgreSQL: <http://www.postgresql.org>
- OpenACS kører under Linux

### Oracle Sekvenser

Med en sekvens i Oracle (*sequence*) kan vi generere en række af unikke løbenumre (id).

```
SQL> create sequence id_seq start with 5;  
Sequence created.
```

```
SQL> select id_seq.nextval from dual;
```

```
NEXTVAL
```

```
-----  
5
```

```
SQL> select id_seq.nextval from dual;
```

```
NEXTVAL
```

```
-----  
6
```

```
SQL> select id_seq.currval from dual;
```

```
CURRVAL
```

```
-----  
6
```

```
SQL> drop sequence id_seq;  
Sequence dropped.
```

Tabellen `dual` er en tom tabel som vi f.eks. kan anvende i vores SQL-kommando når syntaksen dikterer, at vi skal angive en tabel uden egentligt at have brug for tabellen.

### Checking sequence numbers in form variables, file `formvars.tcl`

Hidden form variables such as sequences should never fail, unless the user is hacking the system.

```
proc return_panic_error { err_msg } {  
    set title "Problem with this service"  
    set body "Sorry, but it seems that something is wrong with the  
        supplied values. The error is logged,  
        and the system administrator has been notified."  
    <p>  
        Please try again later."  
    return_page $title $body  
    # Log error, and notify the system administrator  
    exit  
}  
# If form_var is non-existing 0 or negative then it is mal formed.  
proc chk_seq_id {form_var {err_msg ""}} {  
    upvar $form_var seq_id  
    if {[info exists seq_id] || !regexp {[1-9][0-9]*}$ $seq_id} {  
        if {[empty_string $err_msg]} {  
            set err_msg "sequence id $form_var is not valid"  
        }  
        return_panic_error $err_msg  
    }  
    return  
}
```

### More about checking of form variables

— Continued from last weeks lecture...

Examples:

- Checking sequence numbers
- Checking ranges — again... Now with examples!
- Checking enumerations

### Checking a range [a,...b], file `formvars.tcl`

A procedure checking that two form variables a and b forms a range [a,...b], where a < b

```
proc chk_range {form_var1 form_var2} {  
    error_msg "You must enter a <b>range</b> \[a,...,b\  
    upvar $form_var1 first  
    upvar $form_var2 last  
    if {[!info exists first] ||  
        !regexp {[0]([1-9][0-9]*)}$ $first} ||  
        !info exists last} ||  
        !regexp {[0]([1-9][0-9]*)}$ $last} ||  
        ($last - $first < 0)} {  
        return_error $error_msg  
    }  
    return  
}
```

In a script that assumes form variables a and b, denoting an interval, we write:

```
source "/web/nh/www/formvars/formvars.tcl"  
set_form_variables 0  
chk_range a b  
# Now, we can safely use a and b
```

Examples of use:

- In time-tracker system for time intervals
- In coursegrader for grading intervals

### Checking Enumerations, file formvars.tcl

Used when a user is asked to enter or select one item from a list of items.

Examples:

1. Yes, No
2. Jan, Feb, Mar, ..., Dec
3. Very Good, Good, Bad, Very Bad
4. 1,2,3,4,5

In the script we write:

```
source "/web/nh/www/Formvars/Formvars.tcl"
set_form_variables 0

chk_enum enum1 [list "Yes" "No"]
chk_enum enum2 [list "Very Good" "Good" "Bad" "Very Bad"]

# Now, we can safely use the two enumerations enum1 and enum2.
```

### Checking Enumerations, file formvars.tcl

```
proc chk_enum {form_var enums {error_msg ""}} {
    upvar $form_var enum
    if [empty_string $error_msg] {
        if {[info exists enum]} {
            append error_msg "You entered form variable $form_var
                with value $enum.<sp>\n"
        } else {
            append error_msg "Form variable $form_var is missing.<sp>\n"
        }
        append error_msg "You must enter one of the following values:
            <blockquote>\n"
            append error_msg [join $enums " " ]
            append error_msg "\n</blockquote>\n"
        }
        if {[!info exists enum]} {
            return_error $error_msg
        } else {
            foreach e $enums {
                if {[string compare $enum $e] == 0} { return }
            }
            # we did not find a match
            return_error $error_msg
        }
    }
}
```

### Site-design ved anvendelse af procedurer

Et site-design kan holdes konsistent ved at benytte en procedure:

```
proc return_page { title body } {
    ns_return 200 text/html " <html>
    <title>$title</title><body bgcolor=white>
    $body
    <hr>
    <address>
    <a href="mailto:nh@t.edu">nh@t.edu</a>
    </address>
    </body>
    </html> "
}
```

```
proc return_page_with_title { title body } {
    return_page $title "<h2>$title</h2> $body"
}
```

Proceduren return\_page\_with\_title kan nu bruges i alle Tcl-programmer der genererer html-sider:

```
# source the utility procedures
source "/web/mael/www/uttl.tcl"
...
return_page_with_title "About" $text
```

### Introduktion til Øvelse 9

Konstruktion af et klassificeringssystem:

<http://www.it.edu/courses/W2/F2002/Lb/Lb9.html>

- select-funktionalløsten group by
- strukturering — tilstandsdiagram for klassificeringssystemet
- procedure til generering af stjerner.

Eksempel: Statistik af sessions

```
create table sessions (
    email varchar(100),
    clicks integer
);

insert into sessions (email, clicks) values ('nh@t.edu', 5);
insert into sessions (email, clicks) values ('kenneth@t.edu', 3);
insert into sessions (email, clicks) values ('nh@t.edu', 9);
insert into sessions (email, clicks) values ('kenneth@t.edu', 8);
insert into sessions (email, clicks) values ('kenneth@t.edu', 3);
insert into sessions (email, clicks) values ('nh@t.edu', 4);
insert into sessions (email, clicks) values ('kenneth@t.edu', 17);
```

### **select-funktionaliteten group by**

```
column email format a20  
  
select email, count(*) from sessions group by email;  
  
select email, sum(clicks) from sessions group by email;  
  
select email, avg(clicks) from sessions group by email;  
  
select email, max(clicks) from sessions group by email;
```

I øvelsen bruges group\_by til beregning af den gennemsnitlige rating af restauranterne

### **Tiden brugt i opgave C - Webstrukturen - fjerner sig hurtigt ind.**

**Planlæg hvilke form-variabler der skal overføres fra en tilstand til en anden.**

### **Procedure til generering af stjerner**

- Til at generere stjerner benytter vi følgende rekursiv procedure:

```
proc genstars { n } {  
  if { $n == 0 } {  
    return ""  
  } else {  
    return "*[genstars [expr $n - 1]]]"  
  }  
}
```

- Når proceduren kaldes med tallet 0 returneres den tomme streng ("")
- Når proceduren kaldes med et andet tal end 0, f.eks. 4, returneres en streng bestående af en stjerne (\*) sammensat med resultatet af at kalde proceduren med tallet 4-1 = 3.
- Hvordan kan vi ellers skrive proceduren genstars?