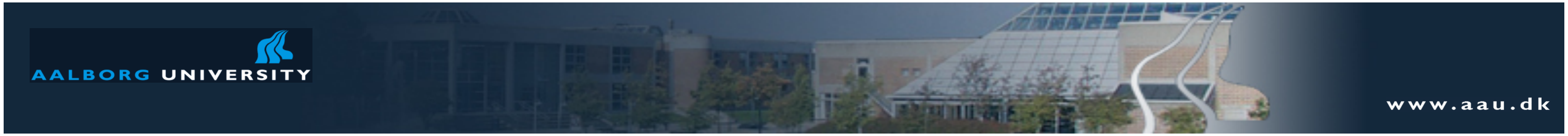
AALBORG UNIVERSITY

Building Informatics

# IKT og Videnrepræsentationer - ICT and Knowledge Representations.

# 8-9. Relational Databases

Cand. Scient. Bygningsinformatik.
Semester 2, 2010.



# CONTENT

- Database history
- Relational database modeling
- Database example
- Installing MySQL, Apache, and PHP
- Structured Query Language, SQL
- Using MySQL

- PHP (ready to lecture 9)
- Web - database connection (ready to lecture 9)

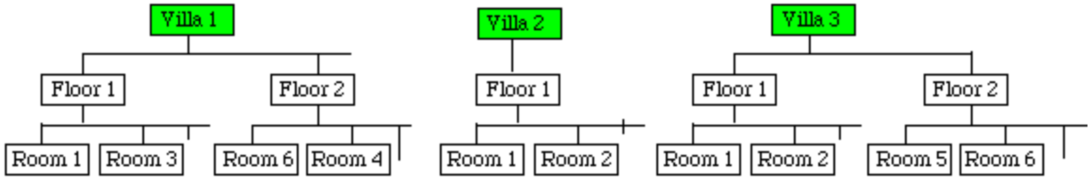
# DATABASE HISTORY

1960 **Files** with records and fields.

- Docuementnb, project, content, date, engineer, department
  122, stibro Aalborg 1997, section, 25-8-1997, Niels Nielsen, engineering 1.

Early 1960. **Hierarchical databases**

- Grows in a tree like structure from the root. Efficient to search but hard to make changes in the structure.

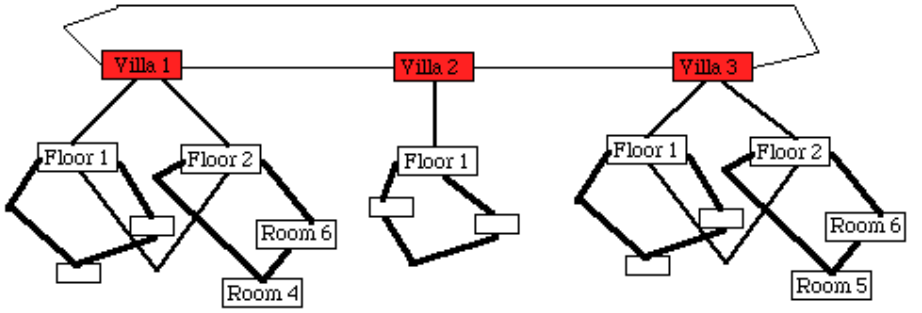


**The Hierarchical Database model**

# DATABASE HISTORY

Mid 1960 The **Network** model

- Uses the association as its basic unit. It consists of linked independent entities. This daabase is more flexible than the hierarchical when it comes to introduce new entities but it may be harder to see through than the hierarchical structure.



**The Network Database model**

# DATABASE HISTORY

## Late 1970 The **Relational Database** Model

- (Codd, 1982), "Relational Database: A Practical Foundation for Productivity". Communications of the ACM (Association for Computing Machinery), February 1982, Volume 25, Number 2.
- Chen Peter Pin-Shan, 1976, "The Entity-Relationship Model - Toward a Unified View of Data".

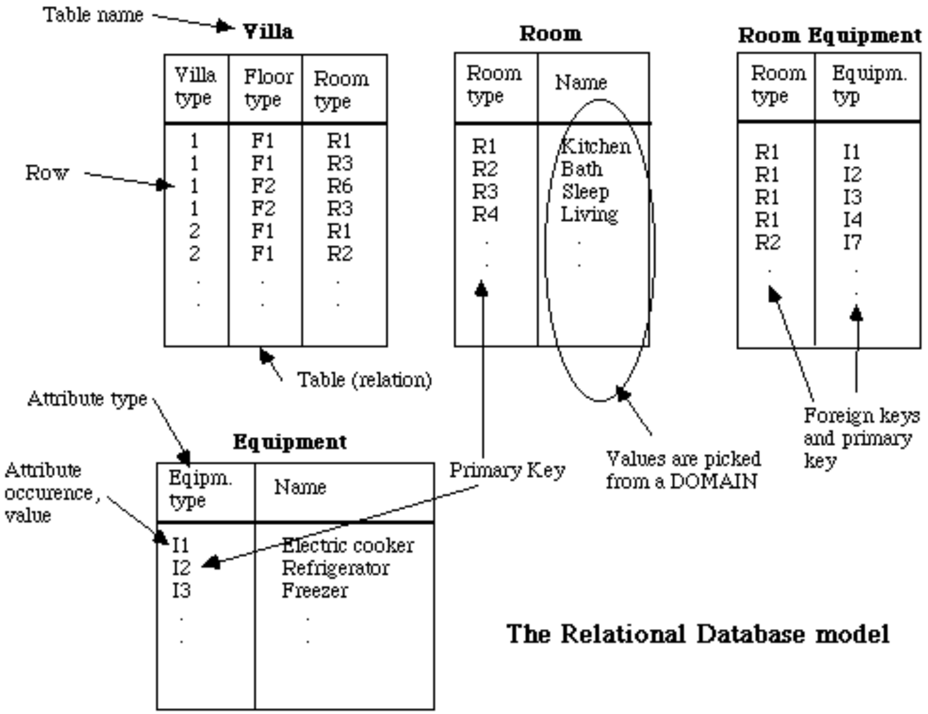


Table name
Row
Table (relation)
Attribute type
Attribute occurence, value
Primary Key
Values are picked from a DOMAIN
Foreign keys and primary key

**Villa**

| Villa type | Floor type | Room type |
|---|---|---|
| 1 | F1 | R1 |
| 1 | F1 | R3 |
| 1 | F2 | R6 |
| 1 | F2 | R3 |
| 2 | F1 | R1 |
| 2 | F1 | R2 |
| . | . | . |
| . | . | . |

**Room**

| Room type | Name |
|---|---|
| R1 | Kitchen |
| R2 | Bath |
| R3 | Sleep |
| R4 | Living |
| . | . |
| . | . |

**Room Equipment**

| Room type | Equipm. typ |
|---|---|
| R1 | I1 |
| R1 | I2 |
| R1 | I3 |
| R1 | I4 |
| R2 | I7 |
| . | . |
| . | . |

**Equipment**

| Eqipm. type | Name |
|---|---|
| I1 | Electric cooker |
| I2 | Refrigerator |
| I3 | Freezer |
| . | . |
| . | . |

**The Relational Database model**

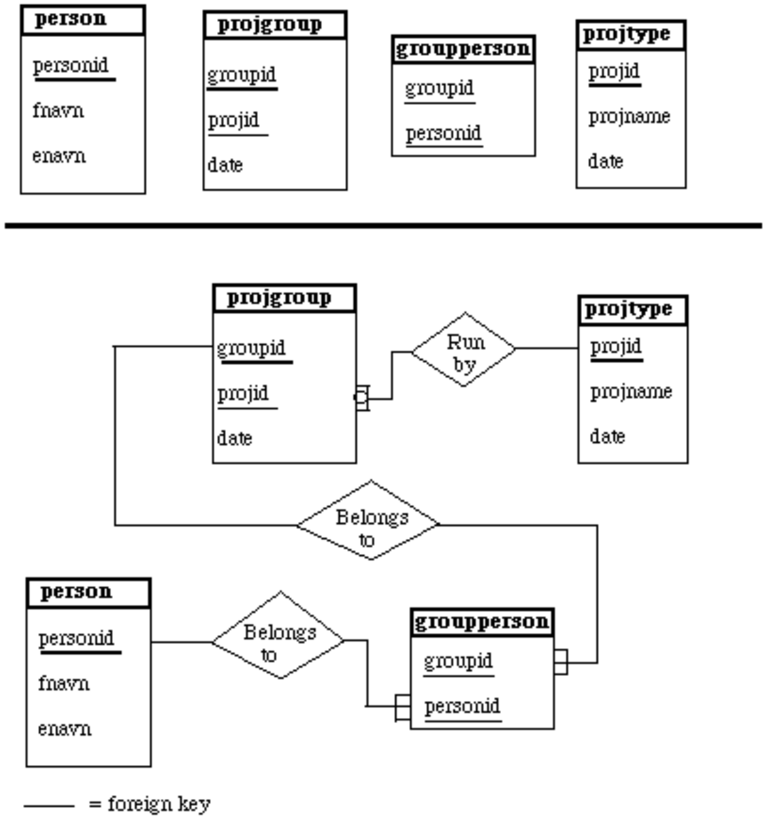
The restrictions to tables are: (Howe D R, 1983, "Data Analysis for Data Base Design". Edward Arnold Ltd., London (page 37)):

- The ordering of rows is not significant; that is, the rows can be interchanged without affecting the information content of the table.
- The ordering of columns is not significant. (We can ensure that this is so by insisting that each column within a table has a distinct attribute type name).
- Each row/column intersection contains a single attribute value. Multiple values are not allowed.
- Each row in a table must be distinct; no two rows can have the same attribute values throughout. (The significance of this rule is that a row can always be uniquely identified by quoting an appropriate combination of attribute values)

# Why use a database?

- Avoid duplication of data (non-redundancy)
- Establish a standardised way to extract adapted and selective data from information containers (both for humans and machines)
- Efficient maintenance, separate information storage from information presentation
- Efficient and easy way to update an information container (stand-alone entities)
- Build information containers available to different programs and web-applications

# The Entity-Relationship model (E-R)



**The sem6_project_db database**

*One-to-one relation:*
(could e.g. be used to link persons one-to-one to different groups of limited set of specified different group types. In this way space could be saved by reducing a number of group type attributes/properties).

*One-to-many relation:*
Example 1: One project type may be run by many different project groups
Example 2: One person may belong to many groups (may be not in semester 6 but anyhow)

*Many-to-many relation:*
Example: Every person may participate in many groups each running different project types. This construct is not allowed therefore we created the 'groupperson'.

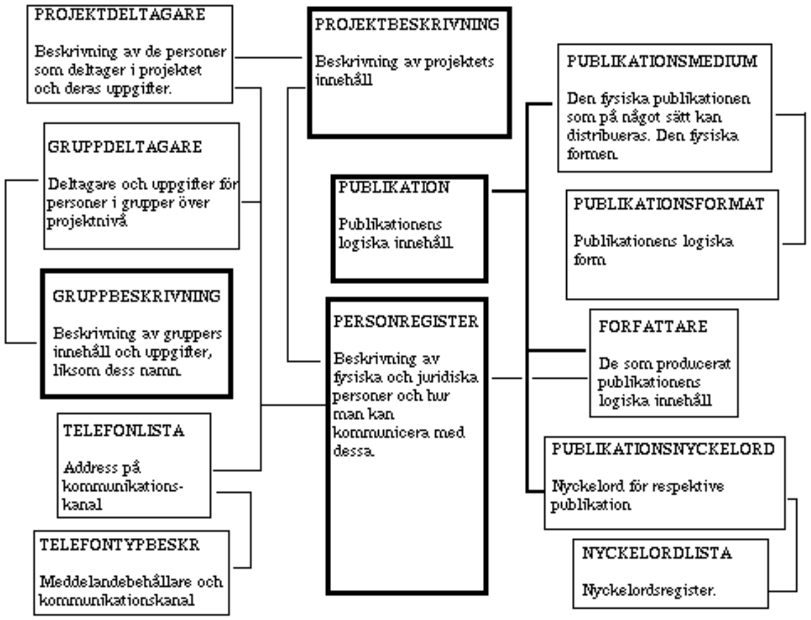
Strong(or regular) entities may be used by other databases (for example 'person') but weak entities containing foreign keys are dependent of the existence of another entity.

The strong relation 'projtype' may have 'projid's which not necessarily are run by any group. This is OK if the entity is strong (which it is). In the figure this is marked by a ring in the fork. As a contrast all persons ('personid') must belong to at lest one group ('groupid').

As before not two primary keys may have the same value or be empty in case it consists of many attributes.

# Database structure example

Projekt Svensk byggforskning på World Wide Web
(Swedish Building Research on the Word Wide Web)
Statens Råd for Bygforskning Sverige (BFR). 1997



PROJEKTDELTAGARE

Beskrivning av de personer som deltager i projektet och deras uppgifter.

GRUPPDELTAGARE

Deltagare och uppgifter för personer i grupper över projektnivå

GRUPPBESKRIVNING

Beskrivning av gruppers innehåll och uppgifter, liksom dess namn.

TELEFONLISTA

Address på kommunikations-kanal

TELEFONTYPBESKR

Meddelandebehållare och kommunikationskanal

PROJEKTBESKRIVNING

Beskrivning av projektets innehåll

PUBLIKATION

Publikationens logiska innehåll

PERSONREGISTER

Beskrivning av fysiska och juridiska personer och hur man kan kommunicera med dessa.

PUBLIKATIONSMEDIUM

Den fysiska publikationen som på något sätt kan distribueras. Den fysiska formen.

PUBLIKATIONSFORMAT

Publikationens logiska form

FORFATTARE

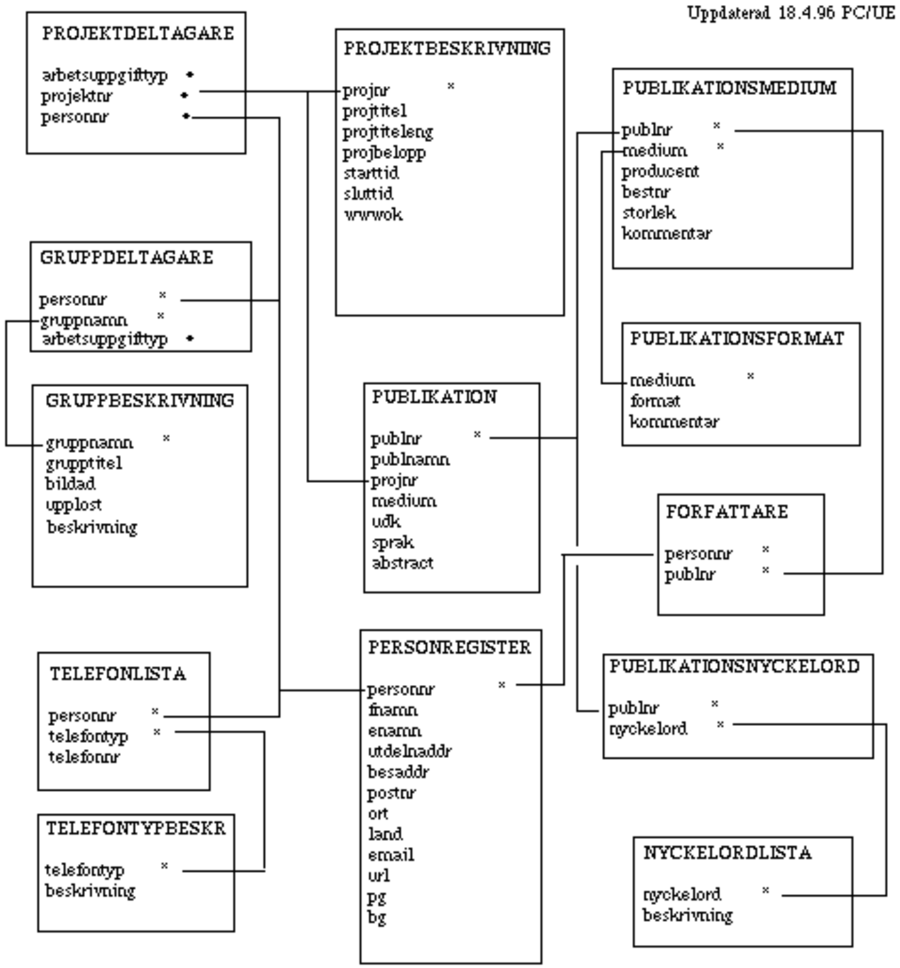
De som producerat publikationens logiska innehåll

PUBLIKATIONSNYCKELORD

Nyckelord för respektive publikation

NYCKELORDLISTA

Nyckelordsregister.



Uppdaterad 18.4.96 PC/UE

PROJEKTDELTAGARE
arbetsuppgifttyp •
projektnr •
personnr •

PROJEKTBESKRIVNING
projnr *
projtitel
projtiteleng
projbelopp
starttid
sluttid
wwwok

PUBLIKATIONSMEDIUM
publnr *
medium *
producent
bestnr
storlek
kommentar

GRUPPDELTAGARE
personnr *
gruppnamn *
arbetsuppgifttyp •

PUBLIKATIONSFORMAT
medium *
format
kommentar

GRUPPBESKRIVNING
gruppnamn *
grupptitel
bildad
upplost
beskrivning

PUBLIKATION
publnr *
publnamn
projnr
medium
udk
sprak
abstract

FORFATTARE
personnr *
publnr *

TELEFONLISTA
personnr *
telefontyp *
telefonnr

PERSONREGISTER
personnr *
fnamn
enamn
utdelnaddr
besaddr
postnr
ort
land
email
url
pg
bg

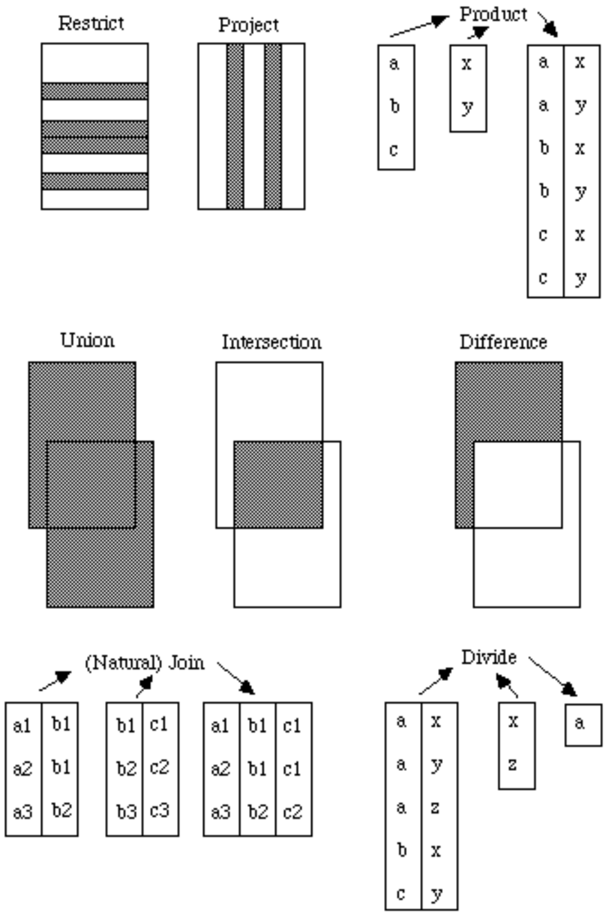
PUBLIKATIONSNYCKELORD
publnr *
nyckelord *

TELEFONTYPBESKR
telefontyp *
beskrivning

NYCKELORDLISTA
nyckelord *
beskrivning

Oversigtlig og detaljeret datamodel over passende fremtidlig lagringsstruktur for forskningsinformation vid BFR.
http://it.civil.aau.dk/it/reports/1997_swebu/r_swebu_3_6_1997.pdf

# Relational Algebra



**Relational algebra operators**

The original eight operators (overview).
[From (Date, 1996) page 140-141].

The relational algebra forms the fundament for the relational databases.

***Restrict:*** Returns a relation consisting of all tuples from a specified relation that satisfy a specified condition (often refereed to as Select. SQL select is though more powerful and includes the functionality of all the eight algebraic operations, and more besides).

***Project:*** Returns a relation consisting of tuples that remain as (sub)tuples in a specified relation after specified attributes have been eliminated

*Product:* Returns a relation consisting of all possible tuples that are a combination of two tuples, one from each of the two specified relations.

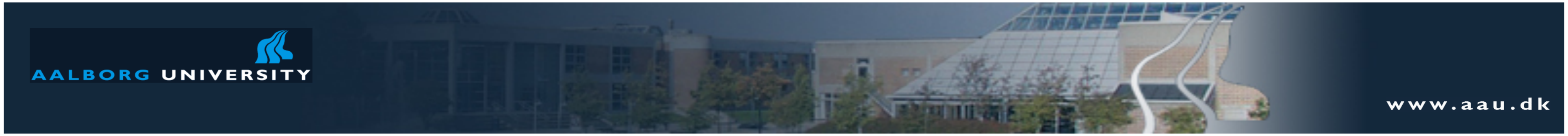
*Union:* Returns a relation consisting of all tuples appearing in either or both of two specified relations.

*Intersect:* Returns a relation consisting of all tuples appearing in both of two specified relations.

*Difference:* Returns a relation consisting of all tuples appearing in the first and not the second of two specified relations.

***Join:*** Returns a relation consisting of all possible tuples that are combination of two tuples, one from each of two specified relations, such that the two tuples contributing to any given combination have a common value for the common attribute(s) of the two relations (and that common value appears just once, not twice, in the result tuple).

*Divide:* Takes two relations, one binary and one unary, and returns a relation consisting of all values of one attribute of the binary relation that match (in the other attribute) all values in the unary relation.



# Normalisation

The normalisation of the logical database layout (conceptual model) will get rid of redundant information and also helps the designers to check the functionality of the database (update rows, delete rows, etc.)

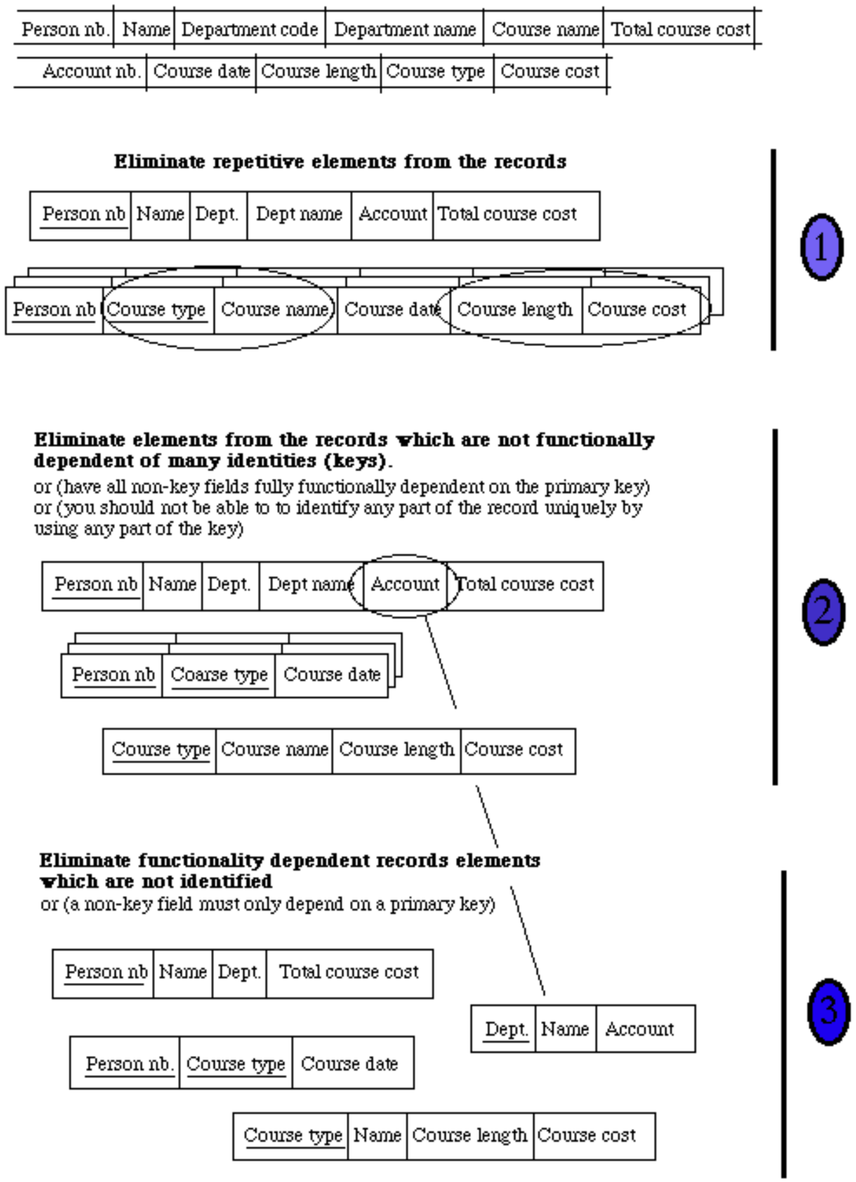
The principle is that we in the conceptual model identify those attributes that identify the rows of a table (entity). We call these attributes primary *key(s).* They are selected from a set of attributes we call candidate keys. Example if two or more persons have the same name it is not enough to use name as primary key. We must then add another candidate key as primary key or instead use a unique personal number as primary key.

The goal is to arrive at a non redundant descriptions of *entities* (thing, person, actions, events, etc.) grouping together properties (*attributes*) that belongs to different objects (later entities in the database), and to provide *unique identifications* for each row (instantiation of an entity).

This process involves mapping and reducing so called functional dependencies between attributes and primary key (or part of it).

We ***normalise*** the model in three (possibly five) steps. First to third normal forms

1. Eliminate repetitive elements from the records. (Basic table structure concern).
2. Elimination of elements from the records which are not functionally dependent of many identities (keys). (Relations between keyed and non-key fields of a record).
3. Elimination of functionally dependent record elements which are not identified. (Relations between non-key fields of records).



| Person nb. | Name | Department code | Department name | Course name | Total course cost |
|---|---|---|---|---|---|

| Account nb. | Course date | Course length | Course type | Course cost |
|---|---|---|---|---|

**Eliminate repetitive elements from the records**

| Person nb | Name | Dept. | Dept name | Account | Total course cost |
|---|---|---|---|---|---|

| Person nb | Course type | Course name | Course date | Course length | Course cost |
|---|---|---|---|---|---|

**Eliminate elements from the records which are not functionally dependent of many identities (keys).**

or (have all non-key fields fully functionally dependent on the primary key)
or (you should not be able to to identify any part of the record uniquely by using any part of the key)

| Person nb | Name | Dept. | Dept name | Account | Total course cost |
|---|---|---|---|---|---|

| Person nb | Coarse type | Course date |
|---|---|---|

| Course type | Course name | Course length | Course cost |
|---|---|---|---|

**Eliminate functionality dependent records elements which are not identified**

or (a non-key field must only depend on a primary key)

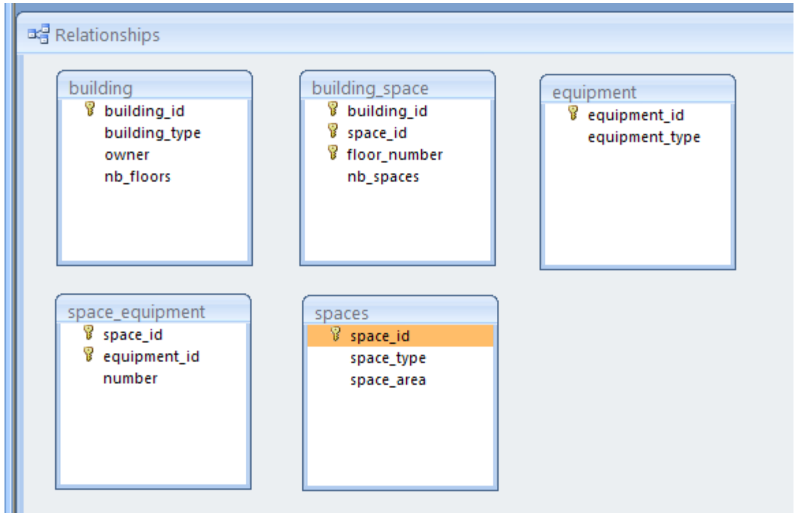
| Person nb | Name | Dept. | Total course cost |
|---|---|---|---|

| Dept. | Name | Account |
|---|---|---|

| Person nb. | Course type | Course date |
|---|---|---|

| Course type | Name | Course length | Course cost |
|---|---|---|---|

# The ‘Building’ database



Relationships

building
- building_id
- building_type
- owner
- nb_floors

building_space
- building_id
- space_id
- floor_number
- nb_spaces

equipment
- equipment_id
- equipment_type

space_equipment
- space_id
- equipment_id
- number

spaces
- space_id
- space_type
- space_area

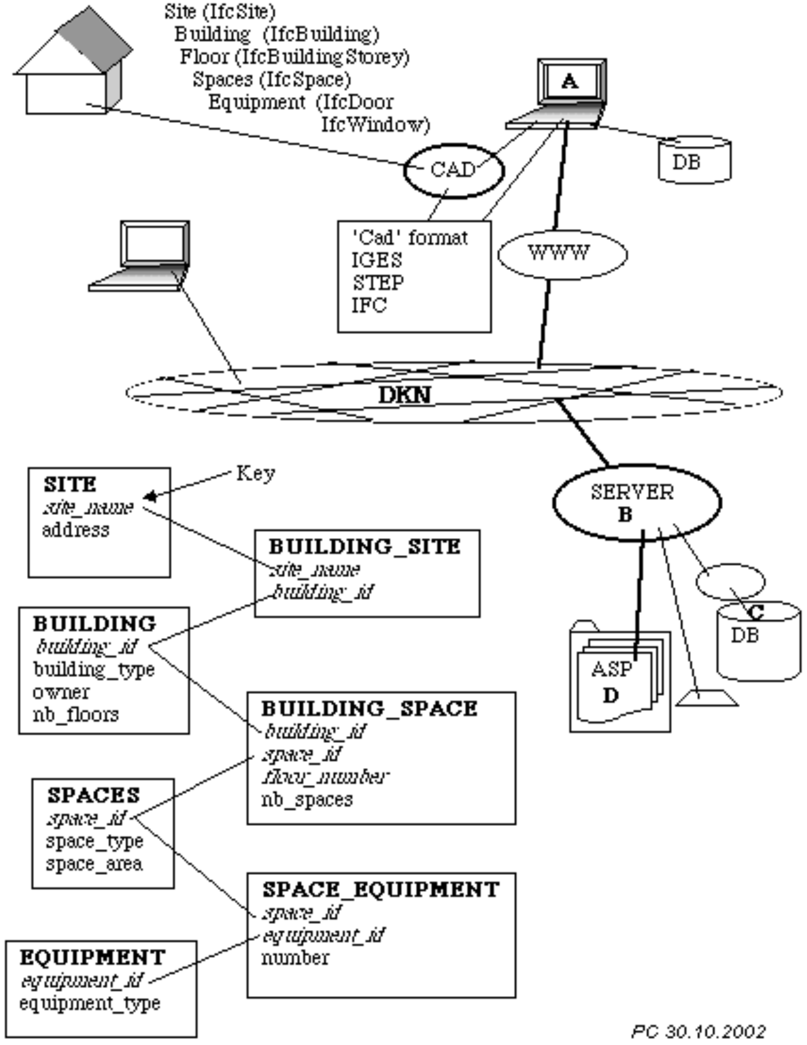
The tables in the building database.
Tables are connected in the SQL WHERE clause ensuring that correct join is made.

The database can be downloaded here

MySQL version
http://it.civil.aau.dk/it/education/models/building_mysql/building.zip
http://it.civil.aau.dk/it/education/models/building_mysql/buildingsem2.sql

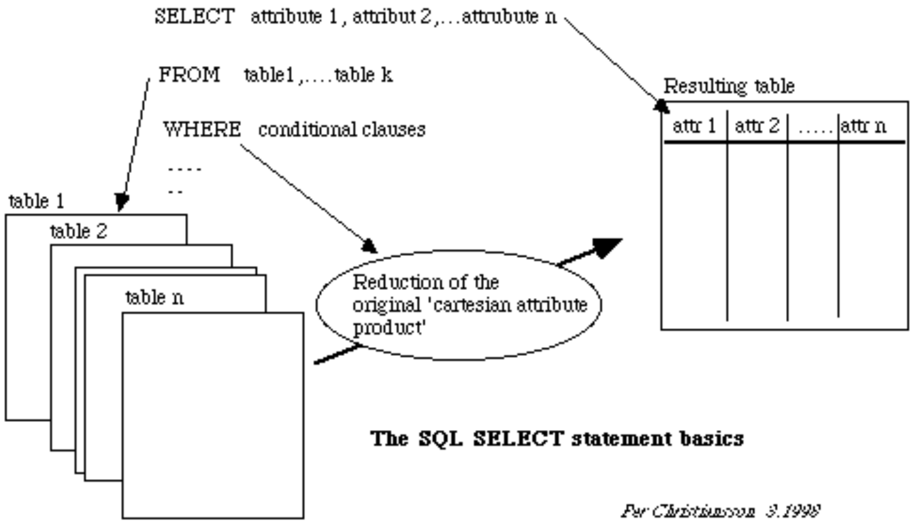
Access version
http://it.civil.aau.dk/it/education/models/building/building.mdb
With ASP example files
http://it.civil.aau.dk/it/education/slides/asp_www_db_building.html

The sem6_project_db database (access),
http://it.civil.aau.dk/it/education/slides/db_sem6_2002_groups.html



PC 30.10.2002

# Structured Query Language

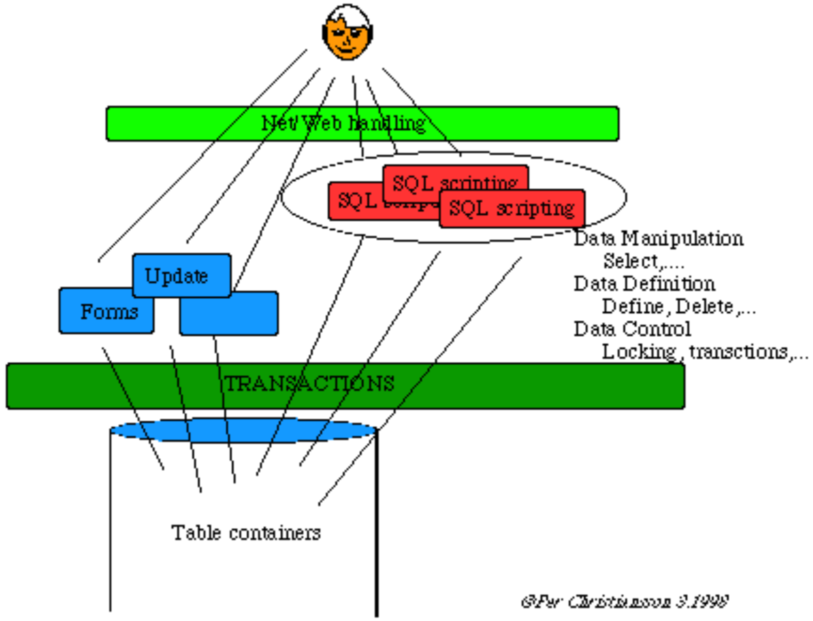

SELECT attribute 1 , attribut 2,...attrubute n
FROM table1 ,....table k
WHERE conditional clauses
....
..
table 1
table 2
table n
Reduction of the original 'cartesian attribute product'
Resulting table
attr 1
attr 2
.....
attr n
The SQL SELECT statement basics
Per Christiansson 3.1998

The SELECT-statement comprises the relational algebra opertors; RESTRICT, PROJECT and JOIN (Date, 1996);

Date C., J., 1996, 1994, "An Introduction to Database Systems". Sixth edition. Addison-Wesley Publishing Company. Reading, Massachusetts. (839 pp.)

| | | | |
|---|---|---|---|
| Relational terms: | relation | tuple | attribute |
| SQL terms: | table | row | column |

# Structured Query Language



©Per Christiansson 3.1998

SQL is a language to control and manipulate relational databases.

- 1979, Relational Software, Inc. (now Oracle) introduced the first commercially available implementation of SQL
- ANSI-86 SQL, SQL89 - SQL/92 - SQL2, SQL99, SQL2003 (xml..), SQL2006, SQL2008
- ANSIX3.135-1992 (600 pages)! (International Standard Database Language)

- SQL.org, http://www.sql.org/
- See also http://www.jcc.com/sql.htm (SQL Standards Home Page)
- W3Schools SQL-tutorial, http://www.w3schools.com/sql/
- http://en.wikipedia.org/wiki/SQL

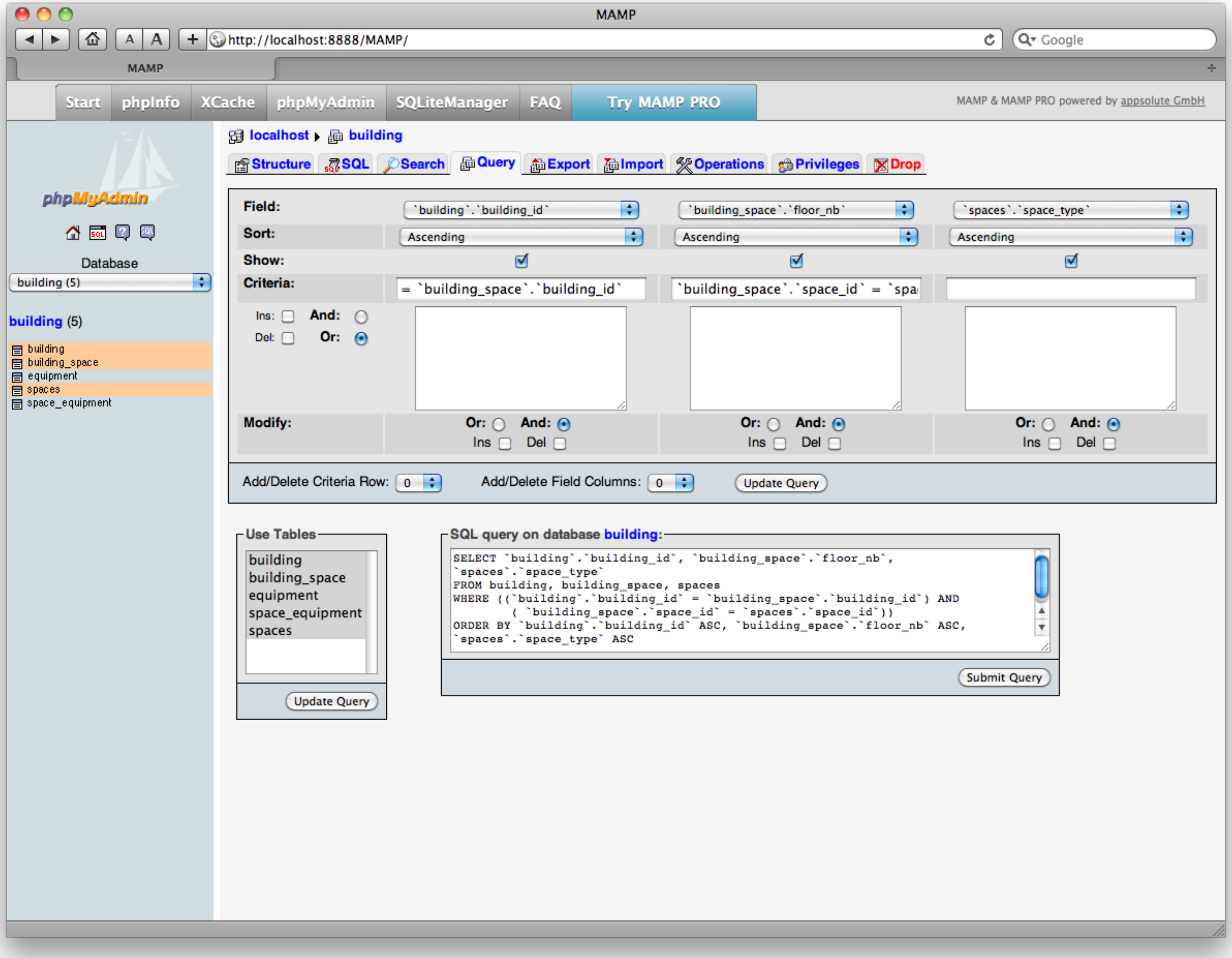
# Installing MYSQL, PHP and APACHE servers

MAMP, http://www.mamp.info/
The abbreviation "MAMP" stands for: Macintosh, Apache, Mysql and PHP. With just a few mouse-clicks, you can install Apache, PHP and MySQL for Mac OS X!

WAMP
http://en.wikipedia.org/wiki/Comparison_of_WAMPs (Comparison of WAMPs)
http://www.wampserver.com/en/

MySQL database
http://dev.mysql.com/downloads/
http://www.phpmyadmin.net

# MYSQL query



| | | | |
|---|---|---|---|
| Field: | \`building\`.\`building_id\` | \`building_space\`.\`floor_nb\` | \`spaces\`.\`space_type\` |
| Sort: | Ascending | Ascending | Ascending |
| Show: | ☑ | ☑ | ☑ |
| Criteria: | = \`building_space\`.\`building_id\` | \`building_space\`.\`space_id\` = \`spa | |
| Ins: □ And: ○<br>Del: □ Or: ● | | | |
| Modify: | Or: ○ And: ●<br>Ins □ Del □ | Or: ○ And: ●<br>Ins □ Del □ | Or: ○ And: ●<br>Ins □ Del □ |

Add/Delete Criteria Row: 0 Add/Delete Field Columns: 0 Update Query

Use Tables

building
building_space
equipment
space_equipment
spaces

Update Query

SQL query on database building:

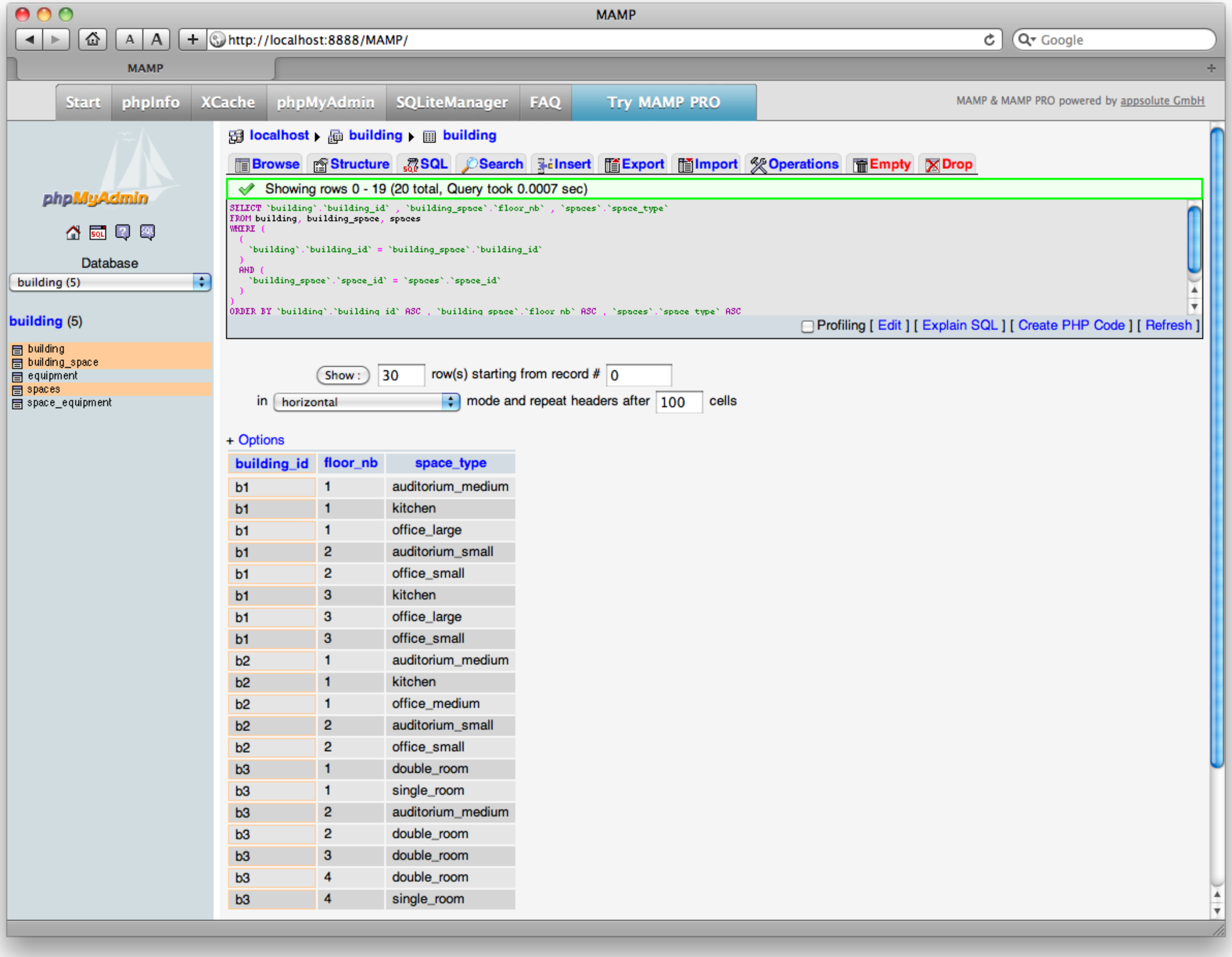
```
SELECT `building`.`building_id`, `building_space`.`floor_nb`,
`spaces`.`space_type`
FROM building, building_space, spaces
WHERE ((`building`.`building_id` = `building_space`.`building_id`) AND
        ( `building_space`.`space_id` = `spaces`.`space_id`))
ORDER BY `building`.`building_id` ASC, `building_space`.`floor_nb` ASC,
`spaces`.`space_type` ASC
```

Submit Query

phpMyAdmin is used as a local database administration program to create and access databases.

You get some help to formulate sql statements. Part of these can later be used in php server side scripting files to access the database from a web-browser.

# MYSQL query



✓ Showing rows 0 - 19 (20 total, Query took 0.0007 sec)

```
SELECT `building`.`building_id` , `building_space`.`floor_nb` , `spaces`.`space_type`
FROM building, building_space, spaces
WHERE (
  (
    `building`.`building_id` = `building_space`.`building_id`
  )
  AND (
    `building_space`.`space_id` = `spaces`.`space_id`
  )
)
ORDER BY `building`.`building_id` ASC , `building_space`.`floor_nb` ASC , `spaces`.`space_type` ASC
```

□ Profiling [ Edit ] [ Explain SQL ] [ Create PHP Code ] [ Refresh ]

Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

+ Options

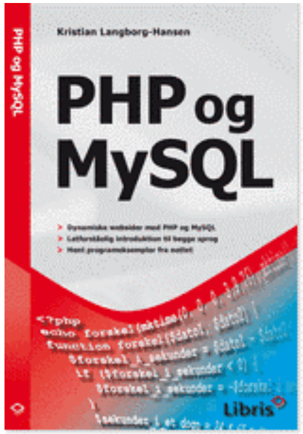
| building_id | floor_nb | space_type |
|---|---|---|
| b1 | 1 | auditorium_medium |
| b1 | 1 | kitchen |
| b1 | 1 | office_large |
| b1 | 2 | auditorium_small |
| b1 | 2 | office_small |
| b1 | 3 | kitchen |
| b1 | 3 | office_large |
| b1 | 3 | office_small |
| b2 | 1 | auditorium_medium |
| b2 | 1 | kitchen |
| b2 | 1 | office_medium |
| b2 | 2 | auditorium_small |
| b2 | 2 | office_small |
| b3 | 1 | double_room |
| b3 | 1 | single_room |
| b3 | 2 | auditorium_medium |
| b3 | 2 | double_room |
| b3 | 3 | double_room |
| b3 | 4 | double_room |
| b3 | 4 | single_room |

This is the result from a query to the database from phpMyAdmin.

# PHP

1994 *Personal Home Page* (Rasmus Lerdorf). Later *PHP: Hypertext Preproessor*

PHP is a server side scripting programming language that can be used as standalone programming language or in combination with HTML.

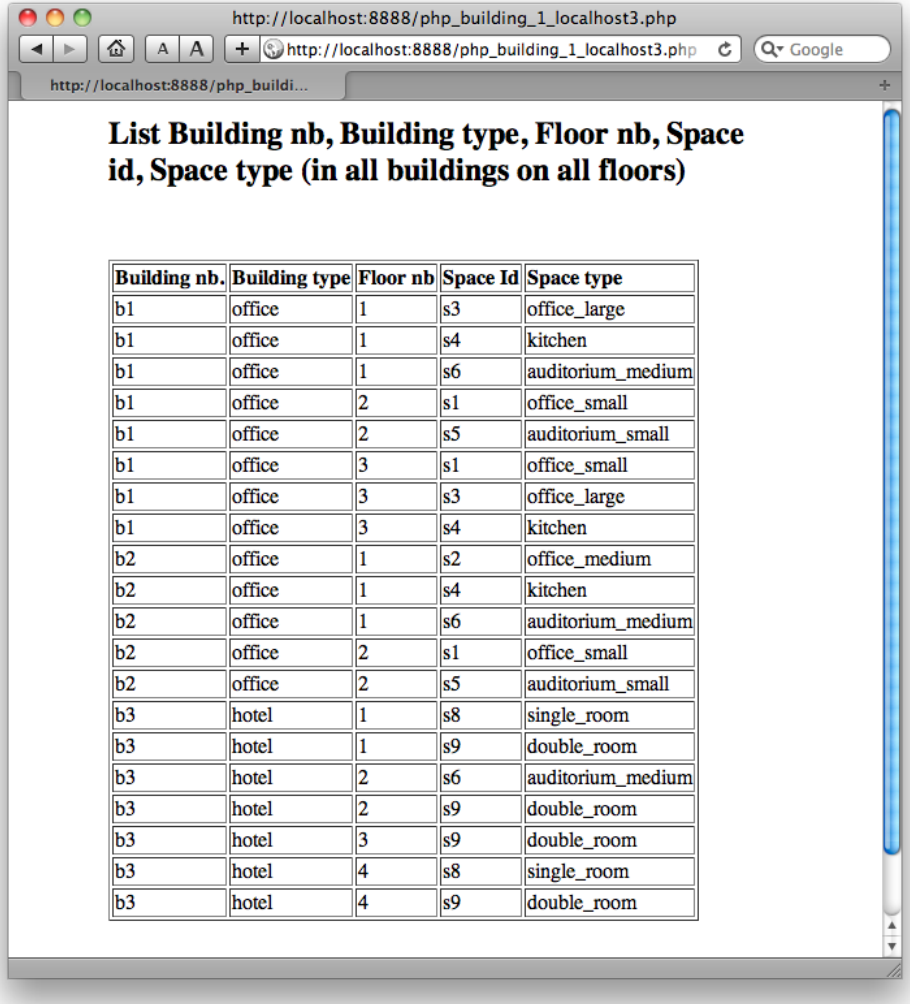


Use /16/ Langborg-Hansen K (2010) "PHP og MySQL". Libris (pp. 84).

Start with chapters 1-6, 8 followed by
Chapters 11, 14, 15.
Use the downloadable PHP-scripting sample example files.

Also use ref
/5/ W3 schools. PHP Basic, PHP Database Tutorials. http://www.w3schools.com/php/
for references.

# Calling MySQL from the WWW



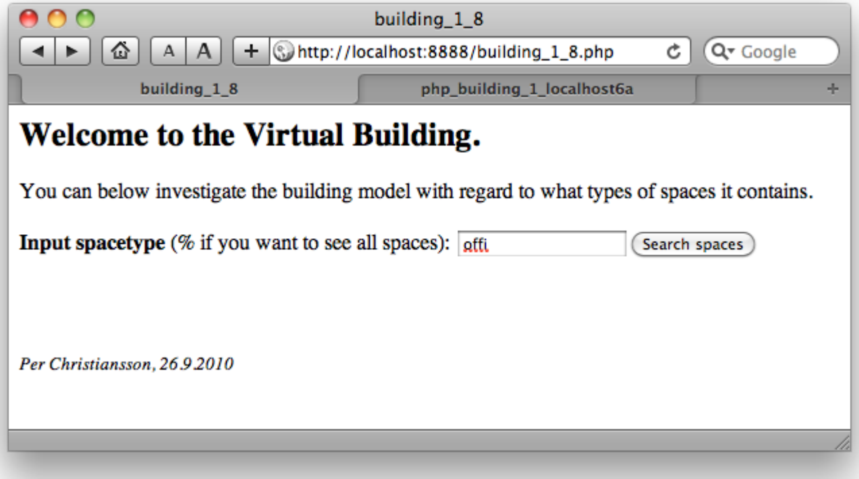
http://localhost:8888/php_building_1_localhost3.php
http://localhost:8888/php_building_1_localhost3.php
Google
http://localhost:8888/php_buildi...

## List Building nb, Building type, Floor nb, Space id, Space type (in all buildings on all floors)

| Building nb. | Building type | Floor nb | Space Id | Space type |
|---|---|---|---|---|
| b1 | office | 1 | s3 | office_large |
| b1 | office | 1 | s4 | kitchen |
| b1 | office | 1 | s6 | auditorium_medium |
| b1 | office | 2 | s1 | office_small |
| b1 | office | 2 | s5 | auditorium_small |
| b1 | office | 3 | s1 | office_small |
| b1 | office | 3 | s3 | office_large |
| b1 | office | 3 | s4 | kitchen |
| b2 | office | 1 | s2 | office_medium |
| b2 | office | 1 | s4 | kitchen |
| b2 | office | 1 | s6 | auditorium_medium |
| b2 | office | 2 | s1 | office_small |
| b2 | office | 2 | s5 | auditorium_small |
| b3 | hotel | 1 | s8 | single_room |
| b3 | hotel | 1 | s9 | double_room |
| b3 | hotel | 2 | s6 | auditorium_medium |
| b3 | hotel | 2 | s9 | double_room |
| b3 | hotel | 3 | s9 | double_room |
| b3 | hotel | 4 | s8 | single_room |
| b3 | hotel | 4 | s9 | double_room |

```
<?php

$con = mysql_connect("localhost:8888","root","root");
if (!$con)
  {
  die('Could not connect: ' . mysql_error());
  }
mysql_select_db("building",$con);

$strQuery = "SELECT `building`.`building_id`,`building`.`building_type`, `building_space`.`floor_nb`,
`building_space`.`space_id`, `spaces`.`space_id`, `spaces`.`space_type` "; $strQuery = $strQuery . "
FROM building, building_space, spaces ";
$strQuery = $strQuery . "
WHERE ((`building`.`building_id` = `building_space`.`building_id`)  AND
(`building_space`.`space_id` = `spaces`.`space_id`) )";
$strQuery = $strQuery . "ORDER BY building.building_id ASC, building_space.floor_nb ASC,
building_space.space_id ASC";

$result = mysql_query($strQuery);
?>
<HTML>
<BODY bgcolor="ffffff">
<h2>List Building nb, Building type, Floor nb, Space id, Space type (in all buildings on all
floors)</h2><BR><BR>
<?php
echo "<table border='1'>";
echo "<tr>";
echo "<td>";
echo "<B>Building nb.</B>" ;
…………………

while($row = mysql_fetch_array($result))
{
…………………
echo "<tr>";
echo "<td>";
echo $row[building_id];
echo "</tr>";
…………
}
echo "</table>";
mysql_close($con);
………….
</BODY>
</HTML>
```

# Calling MySQL from the WWW

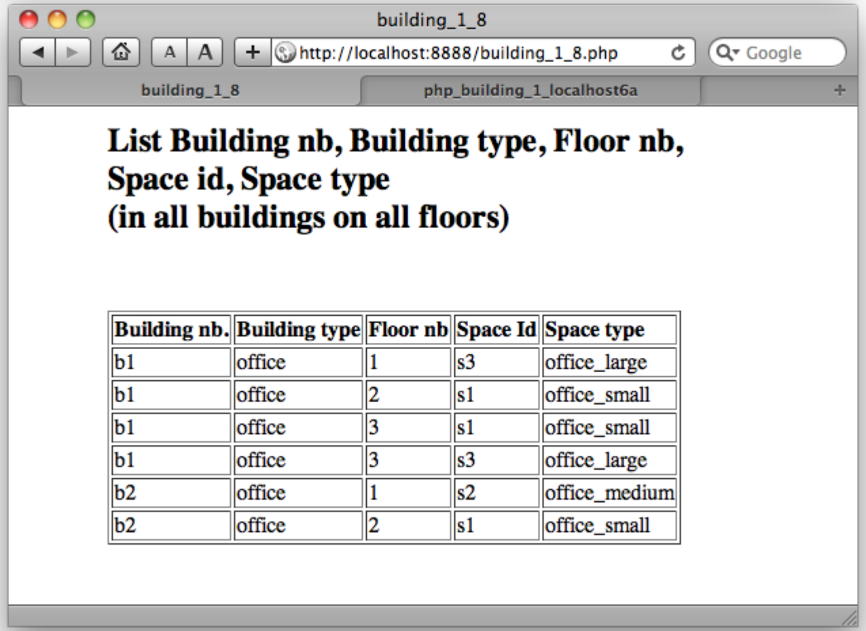

building_1_8
http://localhost:8888/building_1_8.php
Google
building_1_8
php_building_1_localhost6a
Search spaces

## Welcome to the Virtual Building.

You can below investigate the building model with regard to what types of spaces it contains.

**Input spacetype** (% if you want to see all spaces): offi

*Per Christiansson, 26.9.2010*


building_1_8
http://localhost:8888/building_1_8.php
Google
building_1_8
php_building_1_localhost6a

## List Building nb, Building type, Floor nb, Space id, Space type (in all buildings on all floors)

| Building nb. | Building type | Floor nb | Space Id | Space type |
|---|---|---|---|---|
| b1 | office | 1 | s3 | office_large |
| b1 | office | 2 | s1 | office_small |
| b1 | office | 3 | s1 | office_small |
| b1 | office | 3 | s3 | office_large |
| b2 | office | 1 | s2 | office_medium |
| b2 | office | 2 | s1 | office_small |

If you want you can download the  corresponding php file
http://it.civil.aau.dk /it/education/models/building/building_1_8.php

When you develop the server-side php-scripting file in php you should insert comments to explain what the file does. This done in lie with // statement or in blocks surrounded by /*  text    */.

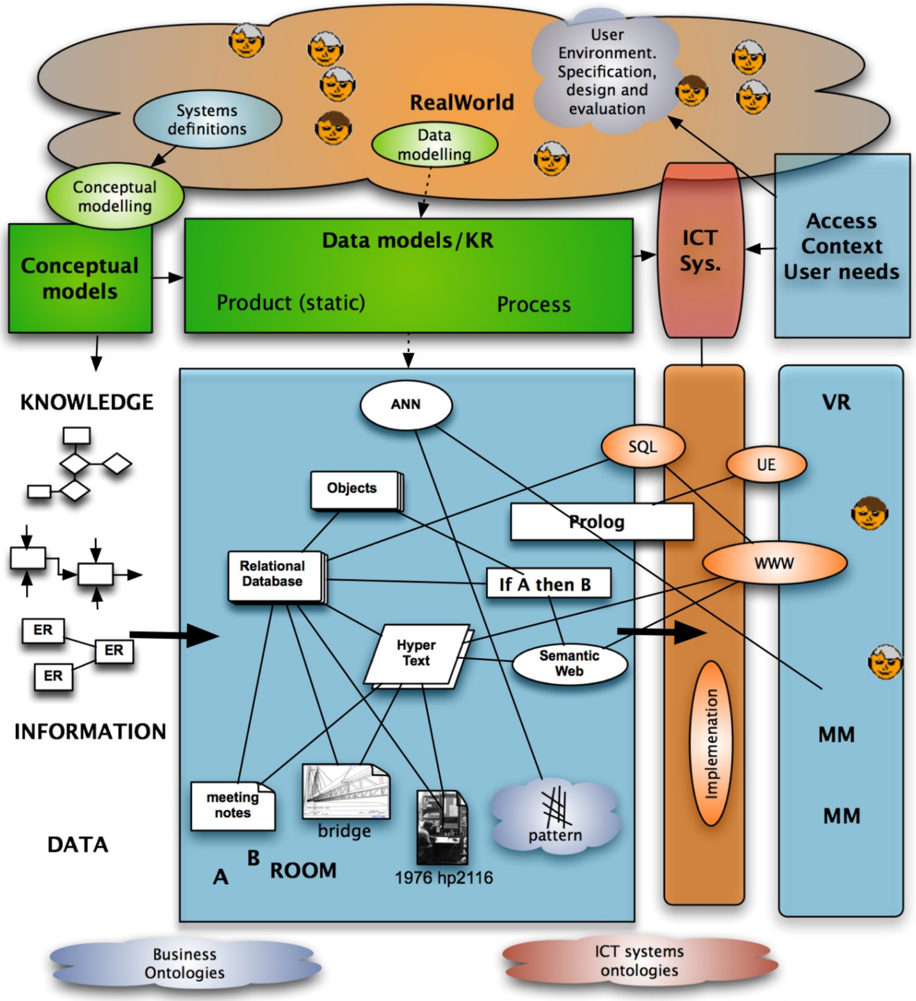
Make progress on small steps running the page on the localhost for each change you make in the php-file. Save versions so you an step back at any time.

In this example we have used if-else statement to be able to handle both forms input and calls to the database in the php-file. Different html files are out put from the server and sent to the web browser depednet on if any search value are given.

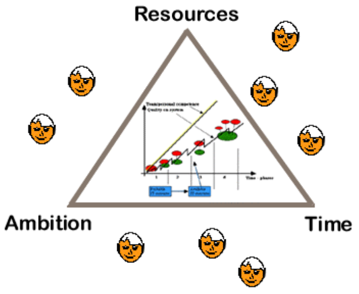
# END

http://it.civil.aau.dk

# SYSTEM DEVELOPMENT



© Per Christiansson 1.10.2007

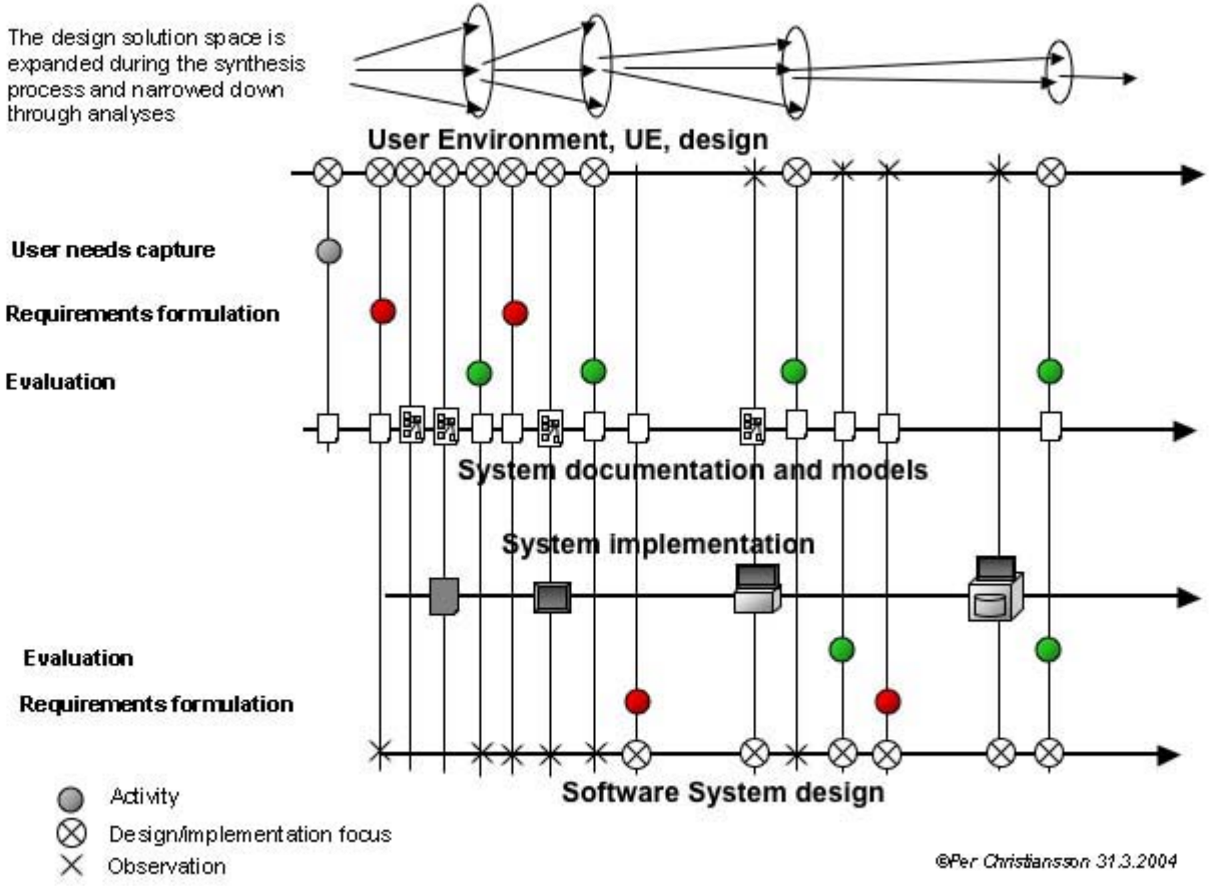


©Per Christiansson 2.2000

Always achieve a good balance between Time, Ambition and Resources.

From the real world to implemented systems in use

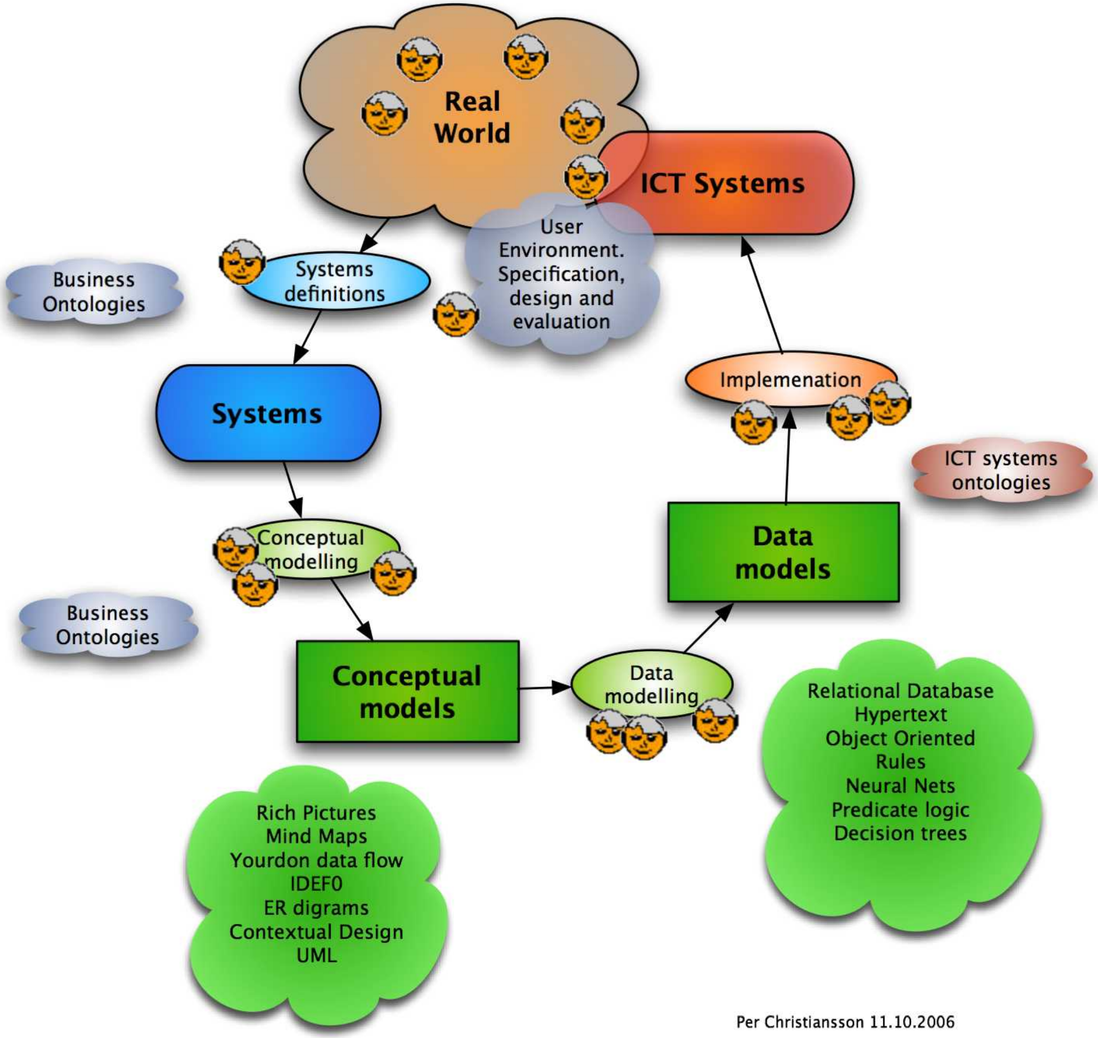
# MODELS OF THE REAL WORLD


The design solution space is expanded during the synthesis process and narrowed down through analyses
User Environment, UE, design
User needs capture
Requirements formulation
Evaluation
System documentation and models
System implementation
Evaluation
Requirements formulation
Software System design
Activity
Design/implementation focus
Observation
©Per Christiansson 31.3.2004

The early design process focuses on user environment, UE, design/implementation
 and the later phases on software development and implementation.

The UE design including user needs capture and user requirements formulations can be supported by contextual design methodology. Different evaluation paradigms can be used as design/implementation progresses.

# SYSTEM DEVELOPMENT



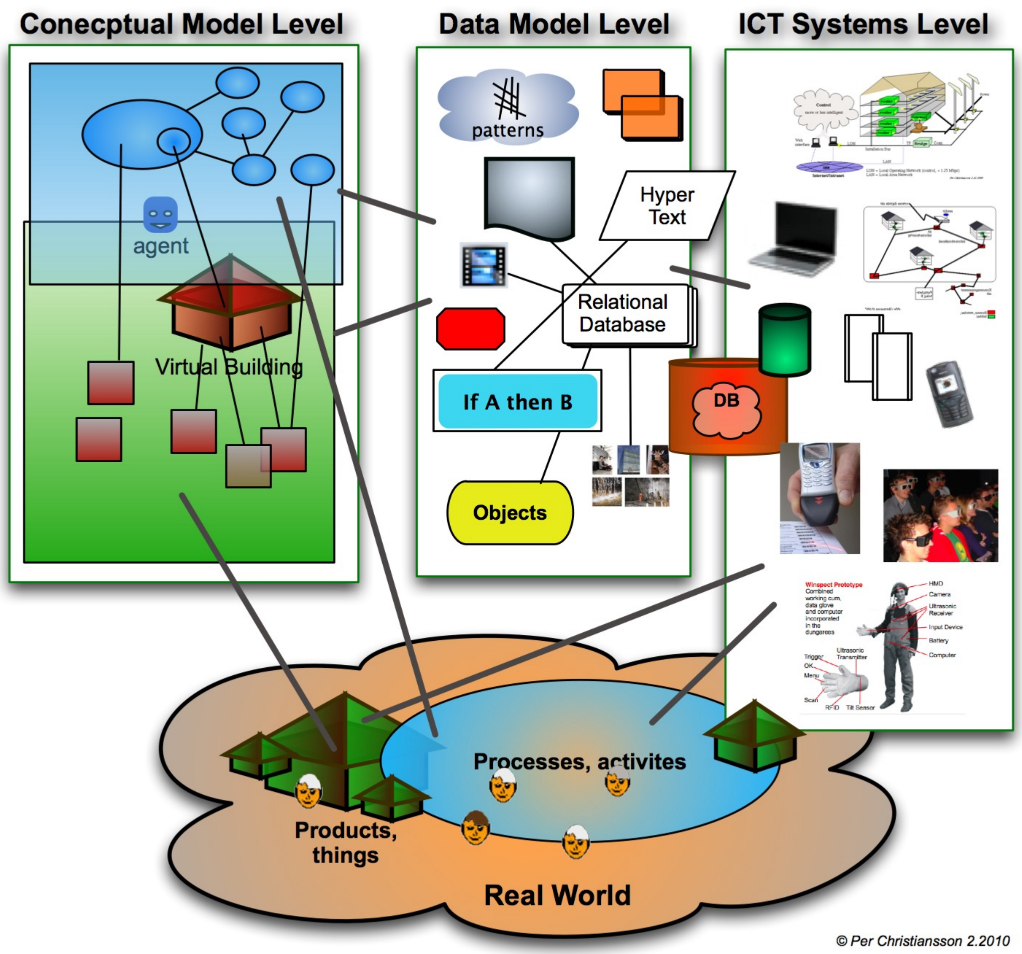
Per Christiansson 11.10.2006

In the real world we identify activities, things, processes, context, and persons.

The real world can be described as (interrelated) systems (no de-facto structure is available today) to accomplish different functions e.g. a comfort system to provide personal living and working quality, personal transport system, load carrying building system, escape system, and communication systems (collaboration, knowledge transfer, mediation, virtual meeting).

# MODELS OF THE REAL WORLD

## The Real World, Models and Systems



© Per Christiansson 2.2010

The HOLISTIC view
The holistic view.

We use different kinds of ICT support in the building process and the built environment.

The ICT systems support different functionalities in the building process and built environment.