Lecture 03: Database components

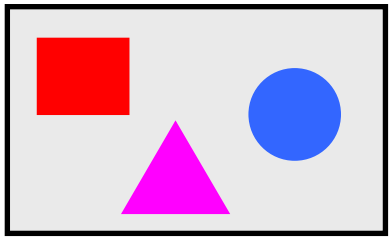
1. ***Database components***
2. ***Objects, operators and integrity rules***
3. ***Database objects***
4. ***The table object***
5. ***Synonyms***
6. ***Schema***
7. ***Integrity rules and constraints: PK, FK, Unique, Check and not null***
8. ***Storage architecture***
9. ***Tablespace and datafiles***
10. ***Conceptual models***
11. ***Object models***
12. ***Data models***

Pär Douhan, pdo@du.se

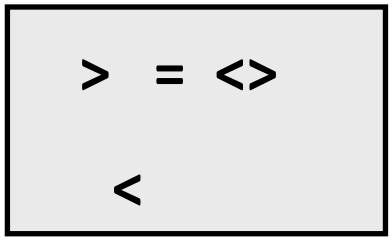
Components

Relational database components

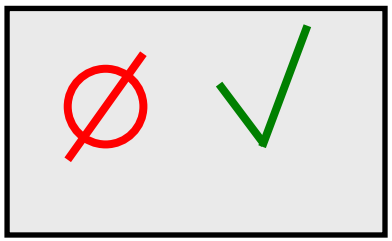
Objects



Operators



Integrity rules



Database objects

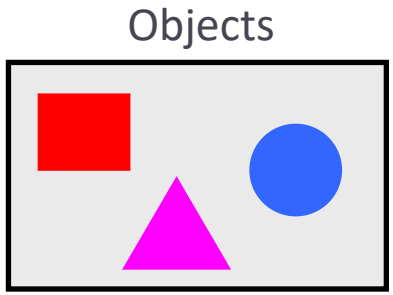


Table
Function
Procedure
Trigger
Sequence
View
Index
User
etc.

How do we create objects?

We do it with SQL DDL.

Example: Create a user:

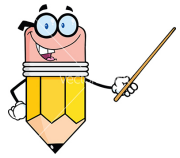
```
create user h19kandr
identified by "123AwRR67"
default tablespace data
temporary tablespace temp
quota 24576 k
on data
account unlock;
```

An empty table

CUSTOMER

cust_id	password	first_name	last_name	email	reg_date	cell

1. Above is an empty table named *customer.*
2. The table has columns that are named.
3. The columns are of a certain data type: number, date or maybe varchar2.
4. The columns can be compared to variables.
5. The table can be compared to a class. In this case a customer class.
6. The table contains no data yet.
7. An **empty table** is equal to **metadata**. Data about the data that will be inserted into the table.

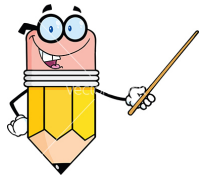


A table with data

CUSTOMER

cust_id	password	first_name	last_name	email	reg_date	cell
56482	OLee45	Rolf	Bjoerk	rb@gmail.com	20190214	+46730211142
89658	PPkkAQW	Malin	Ek	maek@du.se	20141223	
58476	YyYp345	Jossef	Mdoud	jossef@du.se	20130212	+46730533347

1. There are now three rows or objects (= customer objects) in the table.
2. These three rows are the table's data.
3. Data combined with metadata means that the table contains **information**. Information about customers.
4. The smallest element of the table is called a cell.
5. A cell is an intersection of a row and a column.
6. If we look in the column cell for customer with cust_id = 58476.
7. We can see that it says +46730533347 as the value in the cell.

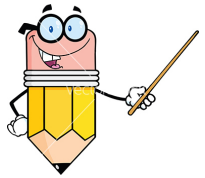


NULL

CUSTOMER

cust_id	password	first_name	last_name	email	reg_date	cell
56482	OLee45	Rolf	Bjoerk	rb@gmail.com	20190214	+46730211142
89658	PPkkAQW	Malin	Ek	maek@du.se	20141223	**NULL**
58476	YyYp345	Jossef	Mdoud	jossef@du.se	20130212	+46730533347

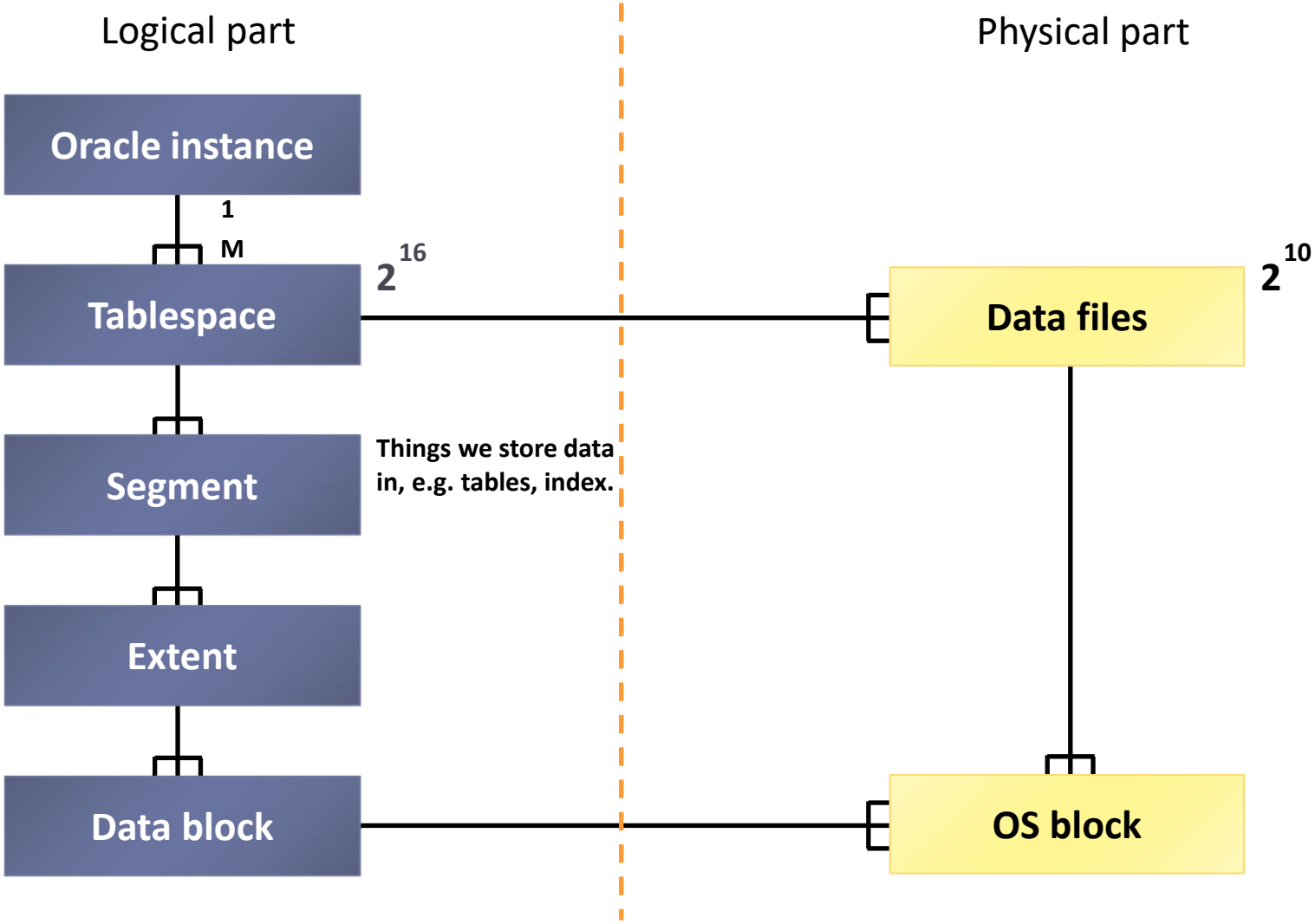
1. Customer with cust_id = 89658 lacks cell phone.
2. The cell is therefore empty.
3. The cell contains a null value.
4. NULL is not the same as 0 (zero).
5. NULL is not defined. Compare it with Mathematics: 1/0 (zero-divide).
6. NULL is not equal NULL.
7. NULL + 100 = ?
8. NULL + 100 is NULL.
9. If we have 100 and add something that is not defined, we get NULL.



Synonyms

swedish	english	"Ted Codd"	the 1960's
tabell	table	relation	file
rad	row	tuple	record
kolumn	column	attribute	field

Storage architecture in Oracle



Search in Data Dictionary

Check what tablespaces and data files that exists by searching the *data dictionary*

```
select tablespace_name, file_name
from dba_data_files
order by tablespace_name;
```

```
TABLESPACE_NAME    FILE_NAME
------------------ ------------------------------------------------------------
SYSAUX             C:\ORACLE12SA\ORADATA\OSRVDB\DATAFILE\O1_MF_SYSAUX_BGCBVOO2_.DBF
SYSTEM             C:\ORACLE12SA\ORADATA\OSRVDB\DATAFILE\O1_MF_SYSTEM_BGCBZ9CT_.DBF
UNDOTBS1           C:\ORACLE12SA\ORADATA\OSRVDB\DATAFILE\O1_MF_UNDOTBS1_BGCC2YR4_.DBF
USERS              C:\ORACLE12SA\ORADATA\OSRVDB\DATAFILE\O1_MF_USERS_BGCC2X5M_.DBF
```

Ensures that new data files ends up in the right place:

```
alter system set DB_CREATE_FILE_DEST = 'C:\ORACLE12SA\ORADATA\OSRVDB\DATAFILE';
```

```
system SET altered.
```

Create a new tablespace

Create tablespace:

```
create tablespace data
extent management local
segment space management auto
online;
```

tablespace data created.

Add data files to the tablespace:

```
alter tablespace data
add datafile 'data01.dbf'
size 1024m;
```

tablespace data altered.

```
alter tablespace data
add datafile 'data02.dbf'
size 1024m;
```

tablespace data altered.

Check!

Check that tablespace and data files were created:

```
select tablespace_name, file_name
from dba_data_files
order by tablespace_name;
```

```
TABLESPACE_NAME    FILE_NAME
------------------ --------------------------------------------------------------------------------
DATA               C:\ORACLE12SA\PRODUCT\12.1.0\DBHOME_1\DATABASE\DATA01.DBF
DATA               C:\ORACLE12SA\PRODUCT\12.1.0\DBHOME_1\DATABASE\DATA02.DBF
DATA               C:\ORACLE12SA\ORADATA\OSRVDB\DATAFILE\OSRVDB\DATAFILE\O1_MF_DATA_BGPD2H1W_.DBF
SYSAUX             C:\ORACLE12SA\ORADATA\OSRVDB\DATAFILE\O1_MF_SYSAUX_BGCBVOO2_.DBF
SYSTEM             C:\ORACLE12SA\ORADATA\OSRVDB\DATAFILE\O1_MF_SYSTEM_BGCBZ9CT_.DBF
UNDOTBS1           C:\ORACLE12SA\ORADATA\OSRVDB\DATAFILE\O1_MF_UNDOTBS1_BGCC2YR4_.DBF
USERS              C:\ORACLE12SA\ORADATA\OSRVDB\DATAFILE\O1_MF_USERS_BGCC2X5M_.DBF
```

It seems ok! We now have 2 data files of 1024 MB each. This provides 2048 MB in total storage space for our tablespace called DATA.

Schema

```
create table customer...
```

user "karin"

```
create table customer...
```

user "erik"

Tablespace data

Segment: karin.customer

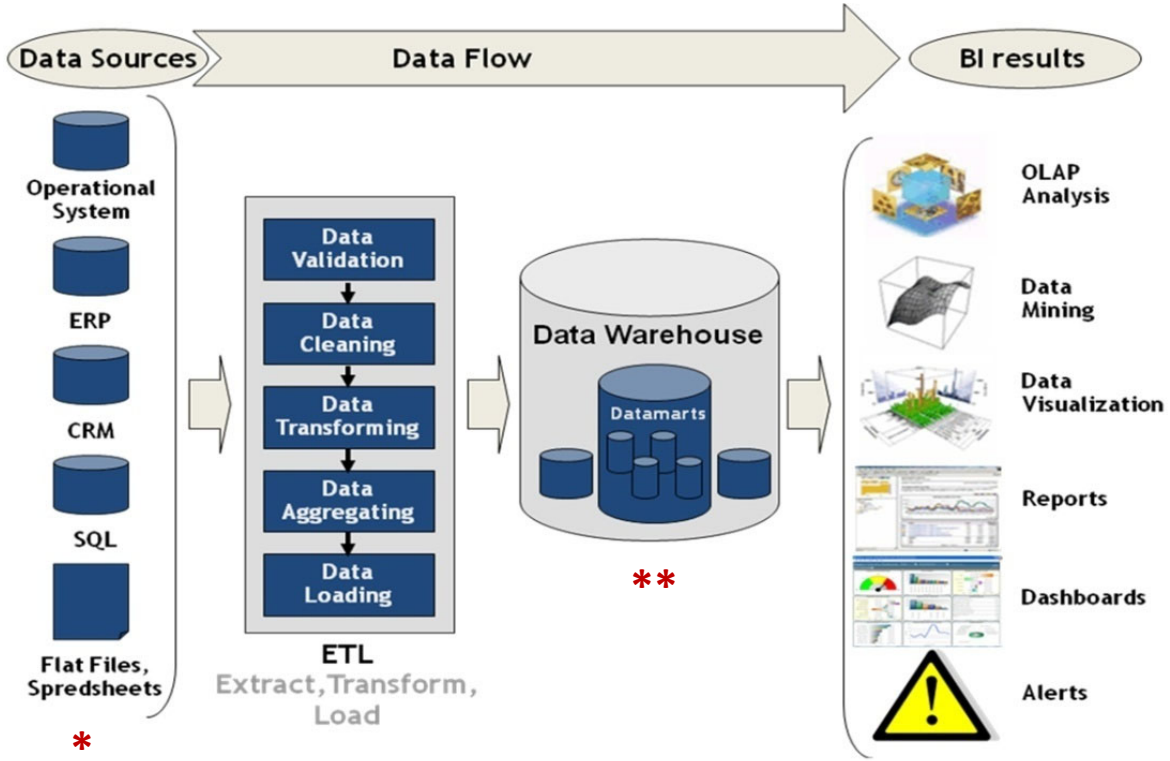
Segment: erik.customer

All items that the user *erik* creates will belong to the schema *erik*.

Integrity rules

Iintegrity rules helps us to:

1. Protect the data in an **operational database** *
2. Help the optimizer in a **Data Warehouse** **



Integrity rules

Integrity rules are maintained with constraints

> "I have a very simple rule: Put dates in dates, numbers in numbers, and strings in strings. Never use a datatype to store something other than what it was designed for, and use the most specific type possible. " (Tom Kyte, Oracle)

Data types are constraints!

Warning for stupidity!

- Varchar2(4000)
- ”a date, hiding in a number, hiding in a string”
- 42 FEB 2014
- Use the right data types!

> "The difference between a question completing in a couple of minutes, or never completing." (Tom Kyte, Oracle)

Constraints

There are **five** different types of constraints for tables:

1. **PK, Primary key**
2. **FK, Foreign key**
3. **Check**
4. **Unique**
5. **Not null**

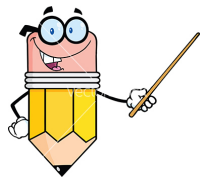
Oracle will store **metadata** about all created constraints in the data dictionary.

Primary key

PK, Primary key

In practice, when choosing or designing **identifying variables**, one can choose between fully constructed, artificial identifications and more or less natural ones.

In any case, it is important that the identifications are **stable at the instance level**. That is that they do not under any circumstances change during the life cycle of the objects (the rows in the table).



Examples of bad constructions are **information-carrying** identifications, where the built-in information is not guaranteed stable (e.g. Swedish social security number).

Examples of good designs are the current registration number for cars, which is **information-free** but still easy to perceive and memorize (e.g. AKM445).

By definition, a value of an identifying variable is a locally unique identifier. Because it uniquely identifies a particular object instance within a given object type (class). (= a unique row in a table)

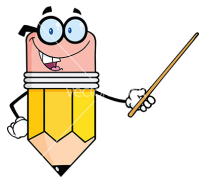
Primary key

CUSTOMER

cust_id	password	first_name	last_name	email	reg_date	cell
56482	OLee45	Rolf	Bjoerk	rb@gmail.com	20190214	+46730211142
89658	PPkkAQW	Malin	Ek	maek@du.se	20141223	
58476	YyYp345	Jossef	Mdoud	jossef@du.se	20130212	+46730533347

We can create a **primary key** and link it to the cust_id column in our customer table. This PK constraint will impose the following limitations:

1. There will **always** be unique values in the column cust_id.
2. There can **never** be two cust_id values that are the same.
3. There will **never** be a null value in the column cust_id.



FK, Foreign key

FK, Foreign key

1. Also called *Referential integrity.*
2. Tables can be related to each other by using a *foreign key constraint.*
3. A **FK** is placed on a column in the *child-table.*
4. As a part of the constraint definition in the child-table, a column is referenced in another table, the *parent-table.*

Parent Table

DEPARTMENT

PK deptno
name

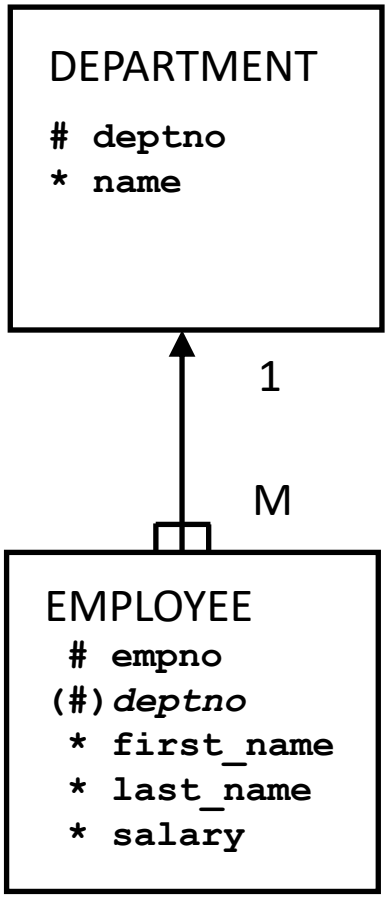
Child Table

EMPLOYEE

PK empno
first_name
last_name
salary
FK deptno

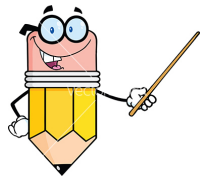
If a row is inserted into the child table, with a non-null value in the FK column, then there must be a row in the parent table that has the same value in the referenced column.

Parent-child or Master-detail



This table can be called:

- *Parent*
- or
- *Master*



This is called **Master-Detail** or **Parent-Child** relationships between tables.

This table can be called:

- *Child*
- or
- *Detail*

FK, Foreign key example

DEPARTMENT

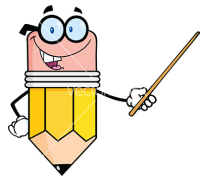
(PK) deptno	name
100	Economy
200	Production
300	Marketing

EMPLOYEE

empno	first_name	last_name	salary	(FK) deptno
125	Olov	Andersson	29000	300
126	Lars	Larsson	34000	300
101	Lena	Ek	35000	200

If we try to insert a new row into the employee table:

```
insert into employee(empno,first_name,last_name,salary,deptno)
values(158,'Arman','Kricic',36500,'400');
```



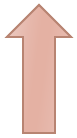
Will this work? **NO!**

Valid values for the FK column EMPLOYEE(deptno) are found in the domain of values that exist for the PK column DEPARTMENT (deptno) i.e. (100, 200, 300) in our example.

Unique

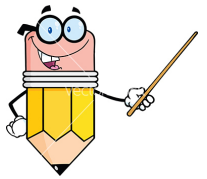
CUSTOMER

cust_id	password	first_name	last_name	email	regdatum	cell
56482	OLee45	Rolf	Bjoerk	rb@gmail.com	20190214	+46730211142
89658	PPkkAQW	Malin	Ek	maek@du.se	20141223	
58476	YyYp345	Jossef	Mdoud	jossef@du.se	20130212	+46730533347



We can create a unique constraint and link it to the email column in our customer table.
This unique constraint will impose the following limitations :

1. There will **always** be unique values in the email column.
2. There can **never** be two email addresses that are the same.
3. There **may be** null values in the email column.
4. This logically follows that null is **not** equal to null.



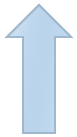
Check

DEPARTMENT

(PK) deptno	name
100	Economy
200	Production
300	Marketing

EMPLOYEE

empno	first_name	last_name	salary	(FK) deptno
125	Olov	Andersson	29000	300
126	Lars	Larsson	34000	300
101	Lena	Ek	35000	200



```
check(salary between 15000 and 150000)
```

We can create a check constraint and attach it to the salary column in our employee table.
This check constraint will impose the following restrictions :

1. There will **always** be salaries in the range between 15 000 and 150 000.
2. It will **not be possible** to give someone a salary of e.g. 12 500 or 185 000.

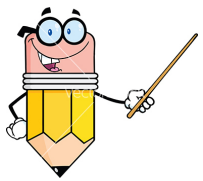
Not null

CUSTOMER

cust_id	password	first_name	last_name	email	reg_date	cell
56482	OLee45	Rolf	Bjoerk	rb@gmail.com	20190214	+46730211142
89658	PPkkAQW	Malin	Ek	maek@du.se	20141223	
58476	YyYp345	Jossef	Mdoud	jossef@du.se	20130212	+46730533347

We can create a **not null** constraint and link it to the email column in our customer table.
This not null constraint will impose the following limitations :

1. There will **never** be a null value in the email column.

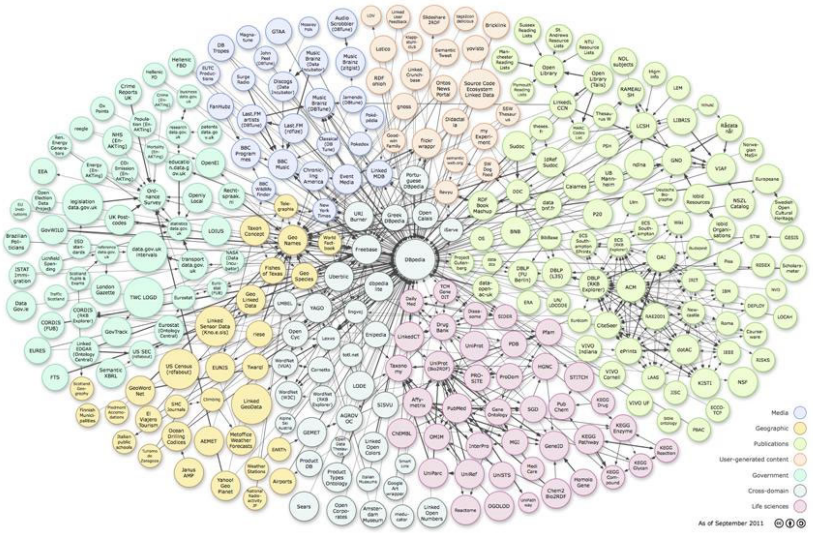


Conceptual models

One should start the *infological* part of the system development work by modeling the piece of reality that IS (the information system) will be about, and the activities that IS will support.

This reality and activity we also call the **object system**.

Object system = Reality of interest = Universe of Discourse



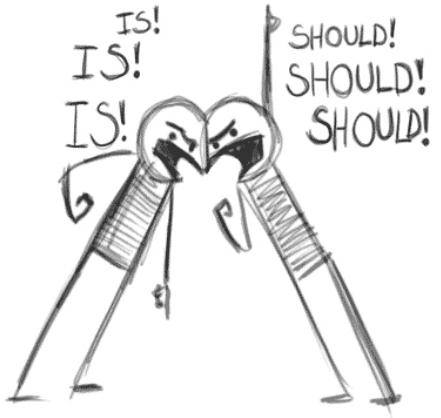
Methods

There is a need for a general conceptual apparatus and a general methodology for modeling a busines, problem area, or simply a piece of reality in a way that is common and understandable to the various stakeholders of an IS.

A model developed for this purpose can be called an *infologically* oriented reality model (an object model).

There are various methods for developing models. At this course you will come into contact with :

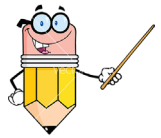
1. **OPR Framework**
2. **Action-oriented conceptual modelling**



Personally, I think it is good to combine these methods as they complement each other.

OPR(t)

We can then transform object models into data models. We can implement data models in a database with SQL DDL (create table).

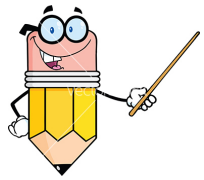


OPR Framework

1. **OPR** stands for *Object-Property-Relationship*.
2. **An object** is a concrete or abstract entity that we are interested in a particular context.
3. **Examples of objects**: *persons, companies, buildings, cars, road accidents, trade transactions, employees, customers, invoices, offers, patients*.
4. **A property** describes an object in a way that is appropriate for a particular purpose. A person can have an age, a home address, an income, etc. Properties can be *characterizing* (general) or *denominating* (identifying).
5. **A relation** is a permanent or temporary association between two or more objects, for example a marriage relation between two persons, an ownership relation between a person and a car, a membership relation between a household and its members, a trade transaction relation between a seller, a buyer, and a product.
6. **Time**, time is found in *infological* models in the form of **moments** and **time intervals**. Examples: An object may have a certain property for a certain time interval. The interest rate on an account can be 2.1% between two moments in time.

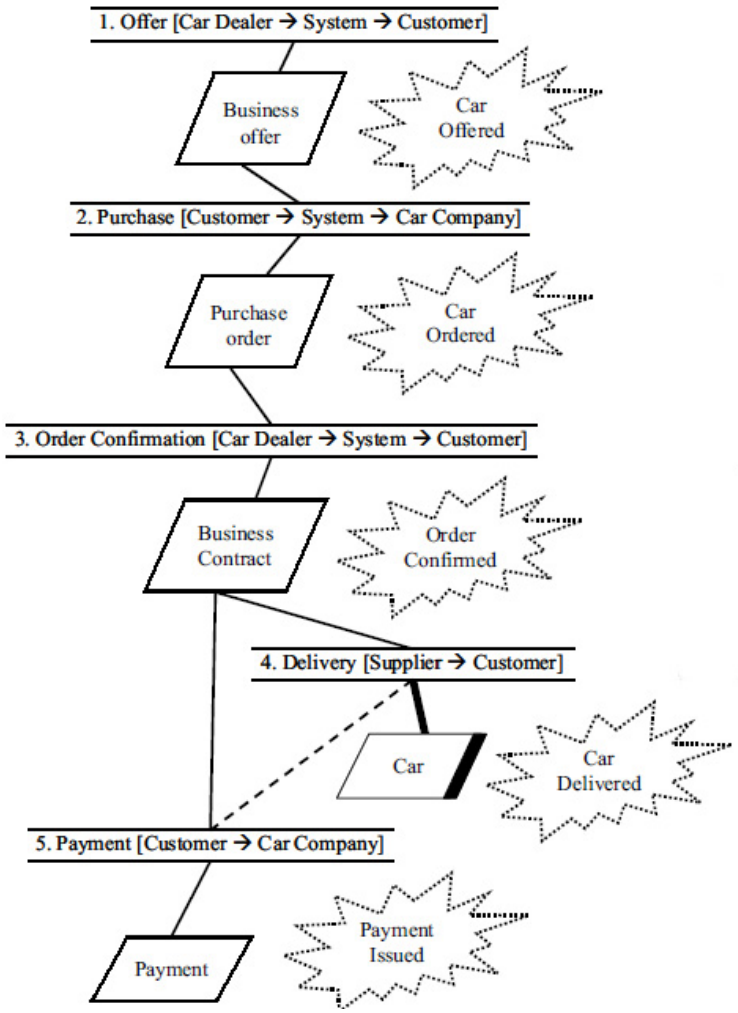
Action-oriented

Action-oriented Conceptual modelling

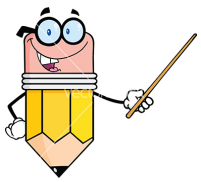


1. In this method, in addition to the objects, **actions** are also focused.
2. Some examples of actions (verb): registrations, orders, confirmations, offers, etc.
3. In the OPR model, one usually talks about **objectifying** a relationship.
4. Here you identify the *action* or the *communicative object* or the *institutional object* directly instead of the relationship.
5. Objects in the form of actors are required to perform different actions.
6. There is a clear link to process models.

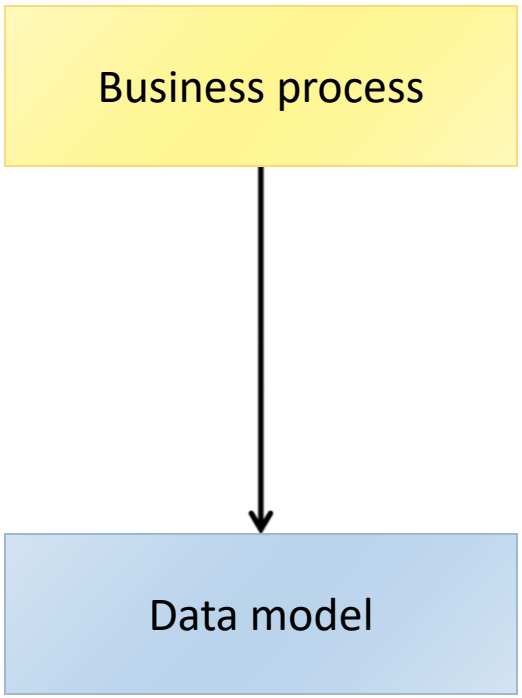
Action-oriented



- In the process model we can directly identify different *communicative objects* (verbs):
- **Business offer (car offerd)**
- **Purchase order (car orderd)**
- **Business contract (order confirmation)**
- **Car deliverd**
- **Payment issued**
- The above objects will be included in our object model.

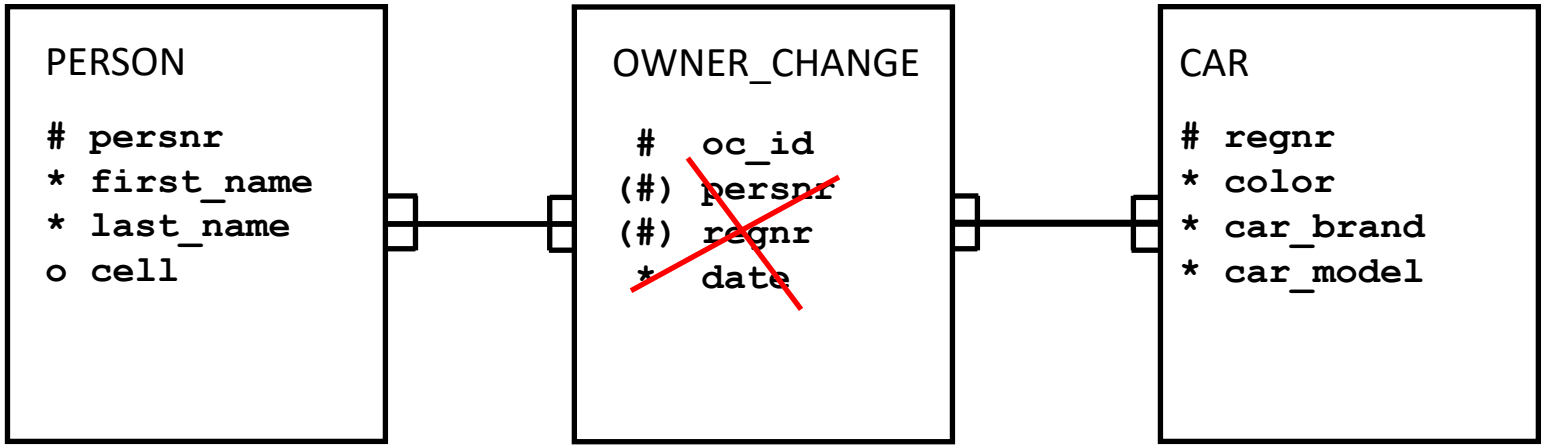


Process – data model

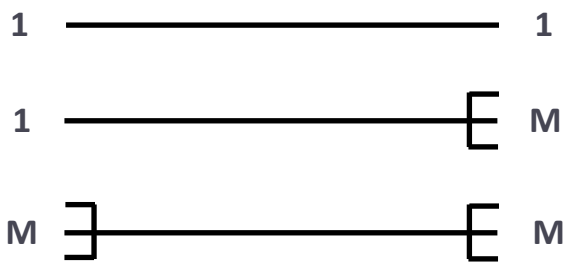


- The data model should provide support for business processes.
- If in the business process, a customer receives a business offer on a car.
- Then that business offer information should be saved in the database.
- If a student signs up for an exam,
- That information should also be saved in the database.
- We have to know the business processes and the business concepts and their relations in order to create a data model.

A simple example model



#	Primary key
(#)	Foreign key
*	Mandatory
o	Optional



PK forms FK in the direction of the fork

A simple example model

In the beginning, it may be good to create **tables with sample data** so that you get an overview and understand the stored information.

PERSON

persnr	first_name	last_name	cell
720217-7415	Carl	Andersson	0730985475
840315-7140	Carina	Larsson	0708365421

CAR

regnr	color	car_brand	model
ABC345	black	volvo	v70
AKM454	blue	audi	a6

OWNER_CHANGE

oc_id	*persnr*	*regnr*	date
24	720217-7415	ABC345	20090201
365	720217-7415	AKM454	20110325
4265	840315-7140	ABC345	20180630