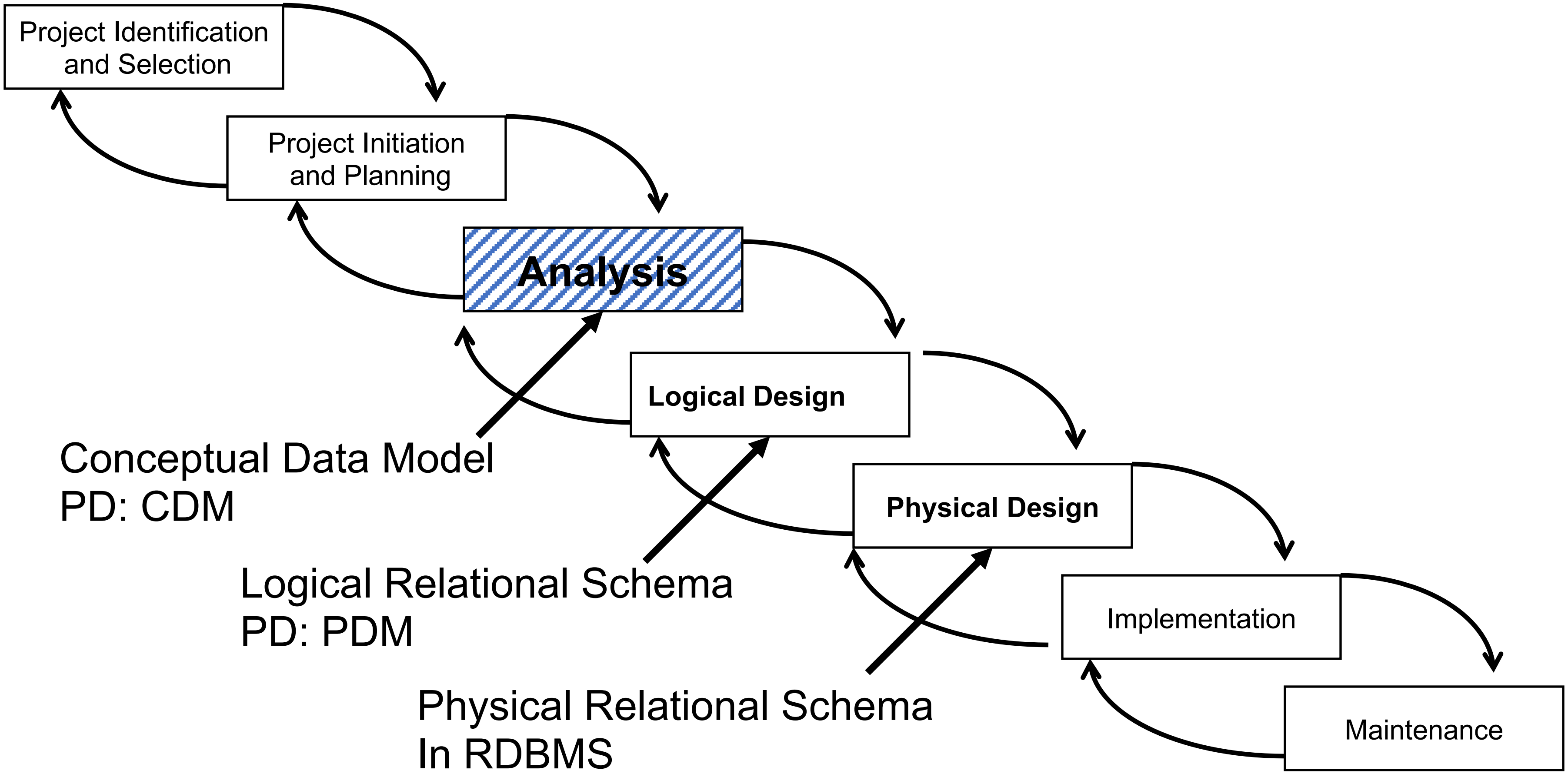
BUILDING AN ER MODEL FROM AN ANALYSIS OF FACTS

HAN_UNIVERSITY OF APPLIED SCIENCES

SOFTWARE LIFE CYCLE REVISITED



Project Identification and Selection

Project Initiation and Planning

Analysis

Logical Design

Physical Design

Implementation

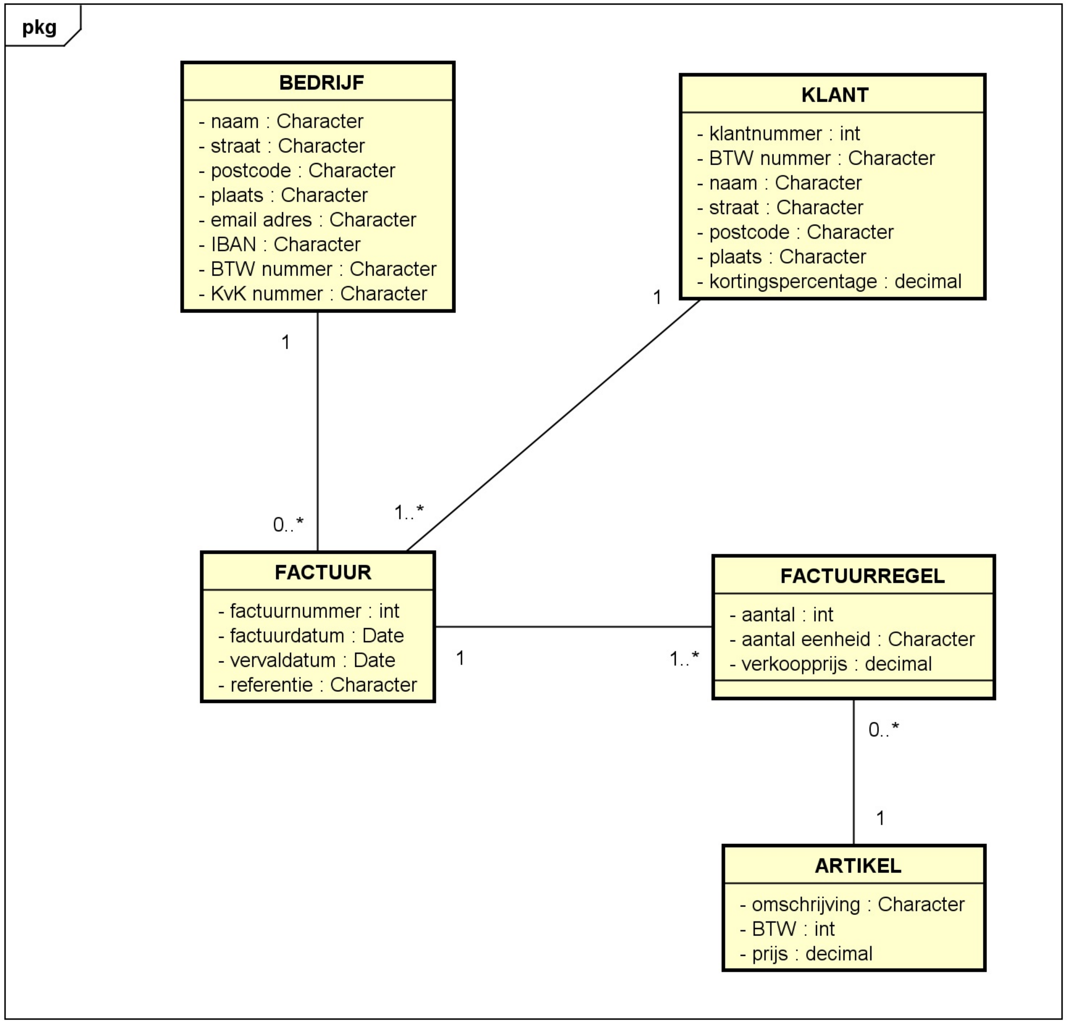
Maintenance

Conceptual Data Model
PD: CDM

Logical Relational Schema
PD: PDM

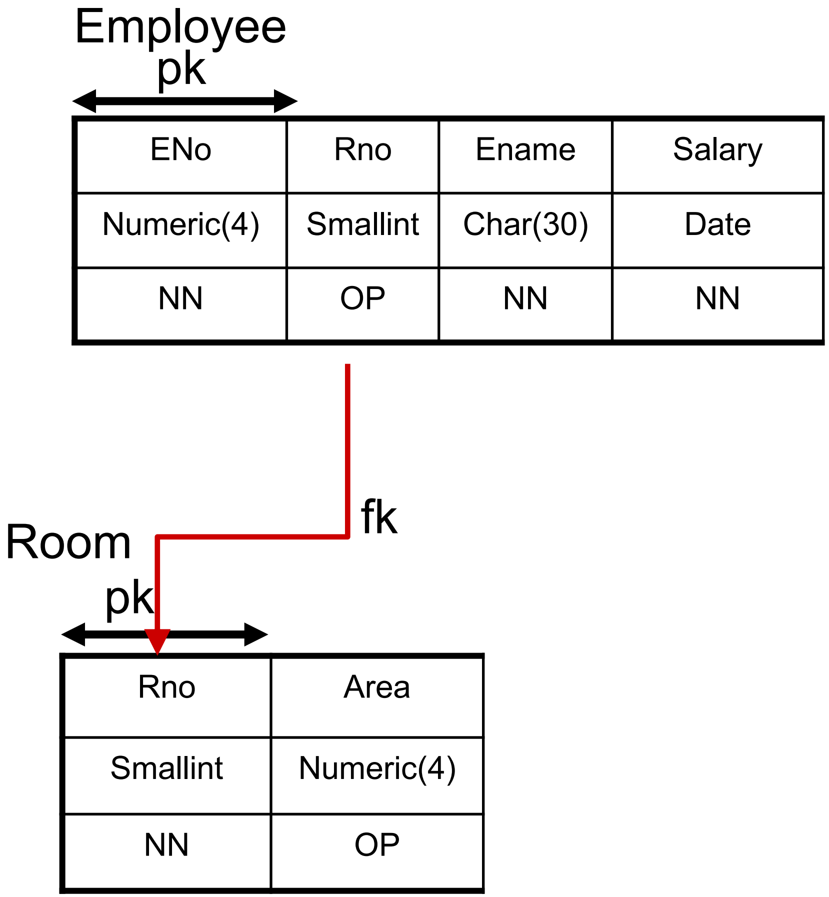
Physical Relational Schema
In RDBMS

FROM SAQ: BUSINESS CLASS DIAGRAM



HAN_UNIVERSITY OF APPLIED SCIENCES

FROM 1ST YEAR DB COURSE:



Employee

pk

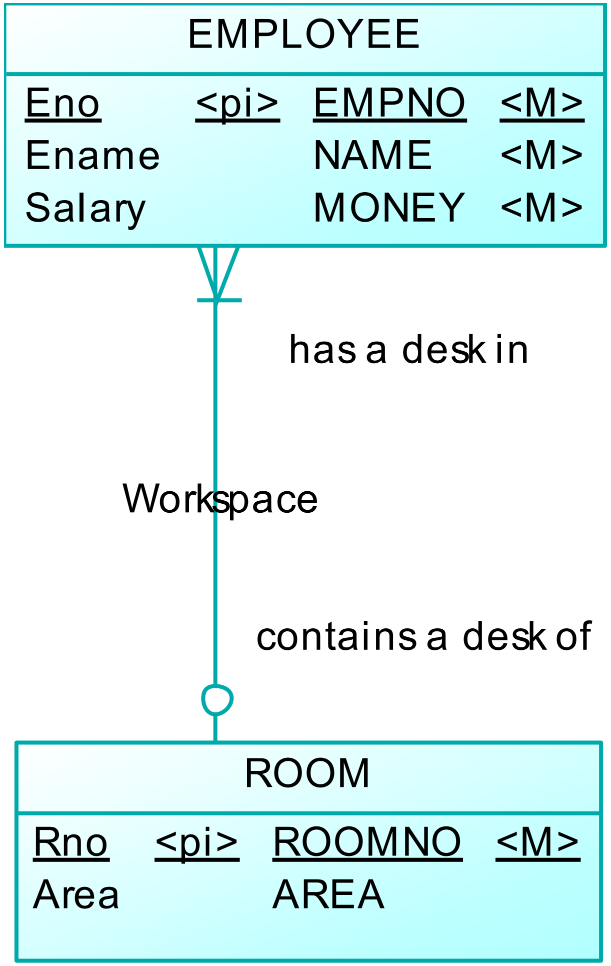
ENo	Rno	Ename	Salary
Numeric(4)	Smallint	Char(30)	Date
NN	OP	NN	NN

Room

pk

Rno	Area
Smallint	Numeric(4)
NN	OP

EXAMPLE ERM DIAGRAM



EMPLOYEE

Eno	<pi>	EMPNO	<M>
Ename		NAME	<M>
Salary		MONEY	<M>

has a desk in

Workspace

contains a desk of

ROOM

Rno	<pi>	ROOMNO	<M>
Area		AREA	

Entity type

Collection of entities: kind of persons / things / concepts about which information is to be stored
Here: EMPLOYEE, ROOM

Attribute

Property of an entity type, defined on a **domain**: the collection of values for this attribute
Here: Eno, Area, etc. on EMPNO, AREA etc.

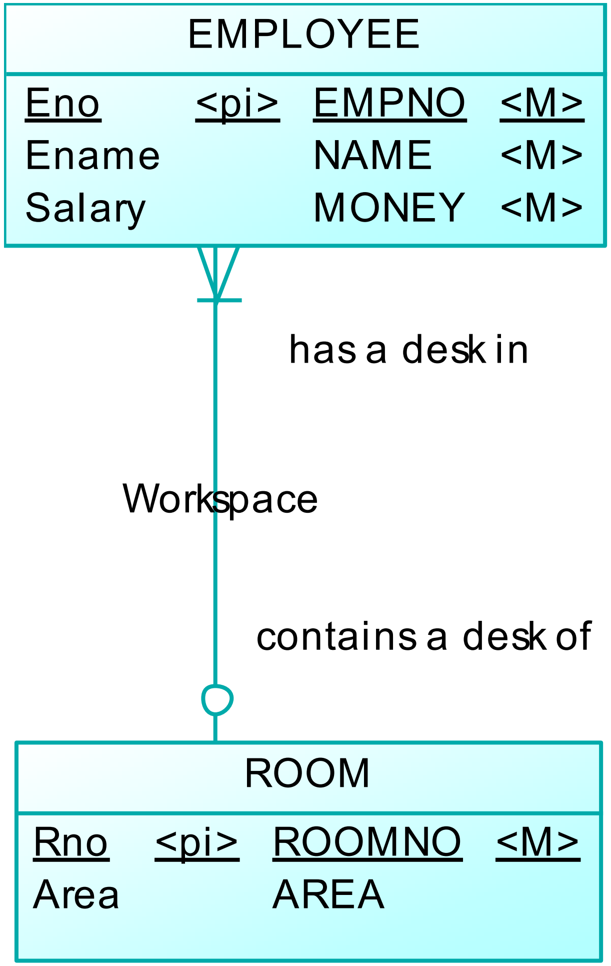
Relationship type

Kind of relationship between entity types
Here: Workspace

SQL SCRIPT

```
/*==============================================================*/
/* Table: EMPLOYEE                                              */
/*==============================================================*/
create table EMPLOYEE (
   ENO                  EMPNO                not null,
   RNO                  ROOMNO               null,
   ENAME                NAME                 not null,
   SALARY               MONEY                not null,
   constraint PK_EMPLOYEE primary key (ENO)
)
/*==============================================================*/
/* Table: ROOM                                                  */
/*==============================================================*/
create table ROOM (
   RNO                  ROOMNO               not null,
   AREA                 AREA                 null,
   constraint PK_ROOM primary key (RNO)
)
alter table EMPLOYEE
   add constraint FK_EMPLOYEE_WORKSPACE_ROOM foreign key (RNO)
      references ROOM (RNO)
```

EXAMPLE ERM DIAGRAM



EMPLOYEE

Eno	<pi>	EMPNO	<M>
Ename		NAME	<M>
Salary		MONEY	<M>

has a desk in

Workspace

contains a desk of

ROOM

Rno	<pi>	ROOMNO	<M>
Area		AREA	

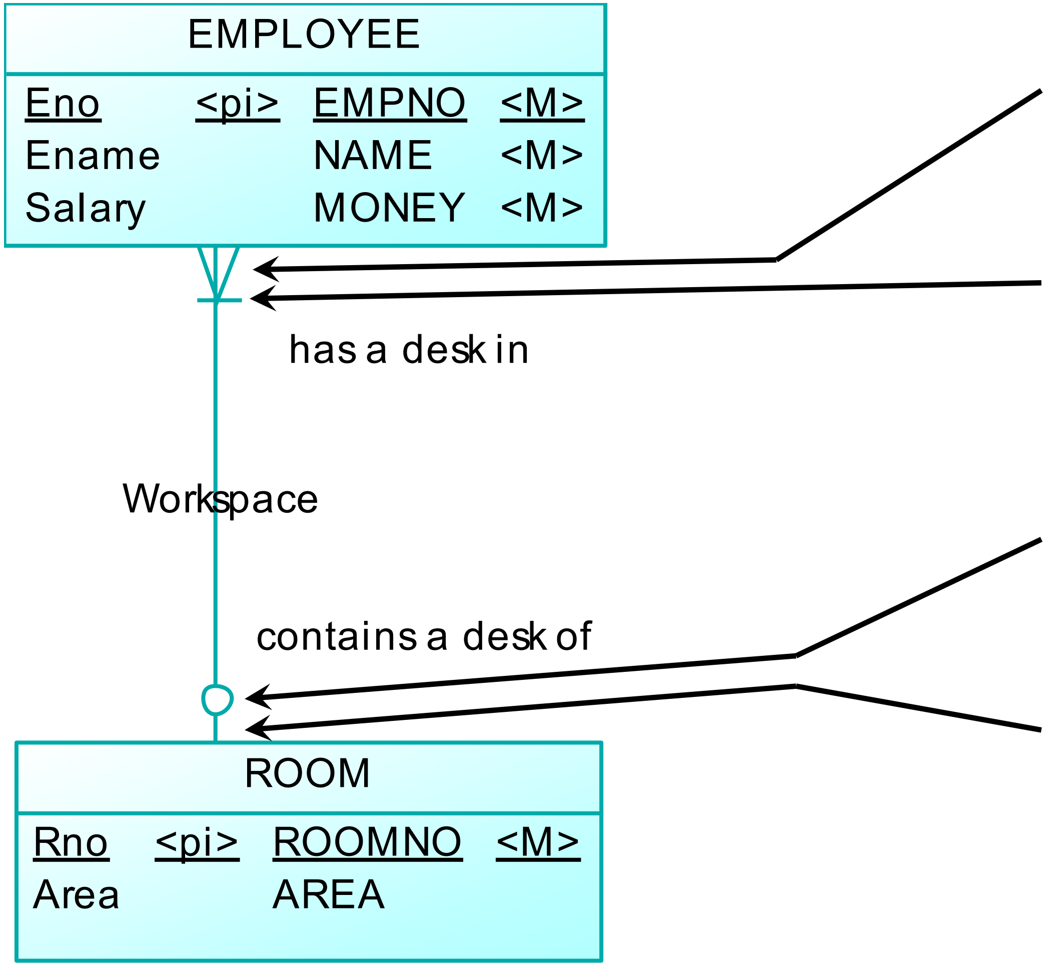
Rule for Entity Type (ET):

Every ET must have a primary identifier: <pi>
Here:
An employee is identified by an Eno
A room is identified by an Rno
Note: there are more complex ways to identify an ET (combinations of Atts, weak ET (slide 6), …

Attribute (Att) mandatory or not:

If a value of an Att must be known for every entity in the ET then the Att is **mandatory:** <M>

CARDINALITIES OF RELATIONSHIP TYPE (RT)



Maximum cardinality: MANY

A room can contain a desk of several employees.

Minimum cardinality: ONE

A room must contain a desk of at least one employee.

Minimum cardinality: ZERO

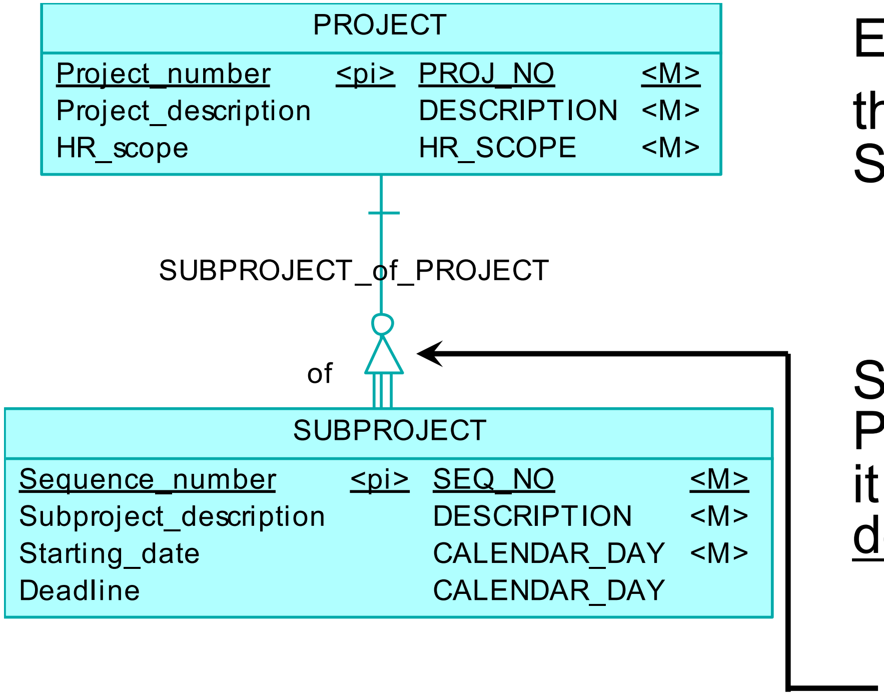
An employee does not need to have a desk in a room.

Maximum cardinality: ONE

An employee can have at most one desk in a room.

NOTE: Max. card. ONE is omitted

WEAK ENTITYTYPE



PROJECT			
Project_number	<pi>	PROJ_NO	<M>
Project_description		DESCRIPTION	<M>
HR_scope		HR_SCOPE	<M>

SUBPROJECT			
Sequence_number	<pi>	SEQ_NO	<M>
Subproject_description		DESCRIPTION	<M>
Starting_date		CALENDAR_DAY	<M>
Deadline		CALENDAR_DAY	

ET PROJECT is identified by
Project_number: ‘project P315’

ET SUBPROJECT is identified by

the combination of
Sequence_number + Project_number:

‘subproject 2 of project P315’

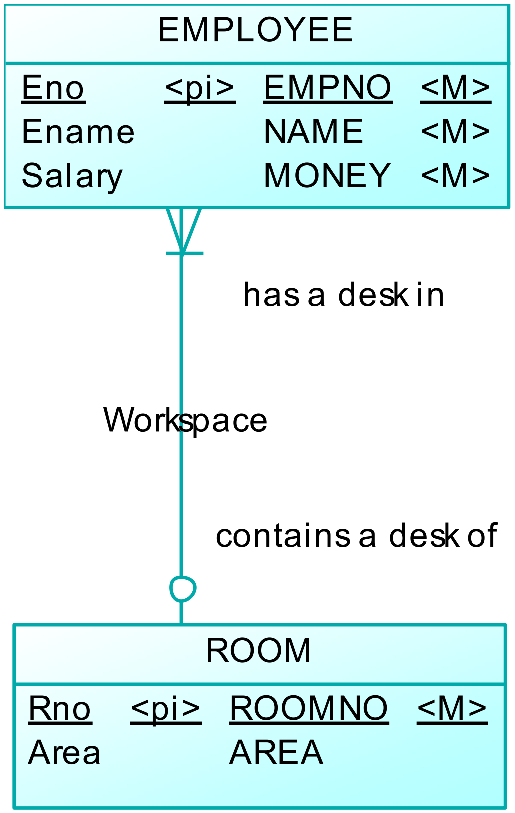
So: SUBPROJECT also needs the <pi> of PROJECT for its own identification:
it is a weak ET (child ET) that is dependent on PROJECT (parent ET)

This is modeled as a RT with a dependency triangle at the child side

HAN_UNIVERSITY
OF APPLIED SCIENCES

TYPES AND INSTANCES

Type level



EMPLOYEE

Eno <pi> EMPNO <M>

Ename NAME <M>

Salary MONEY <M>

has a desk in

Workspace

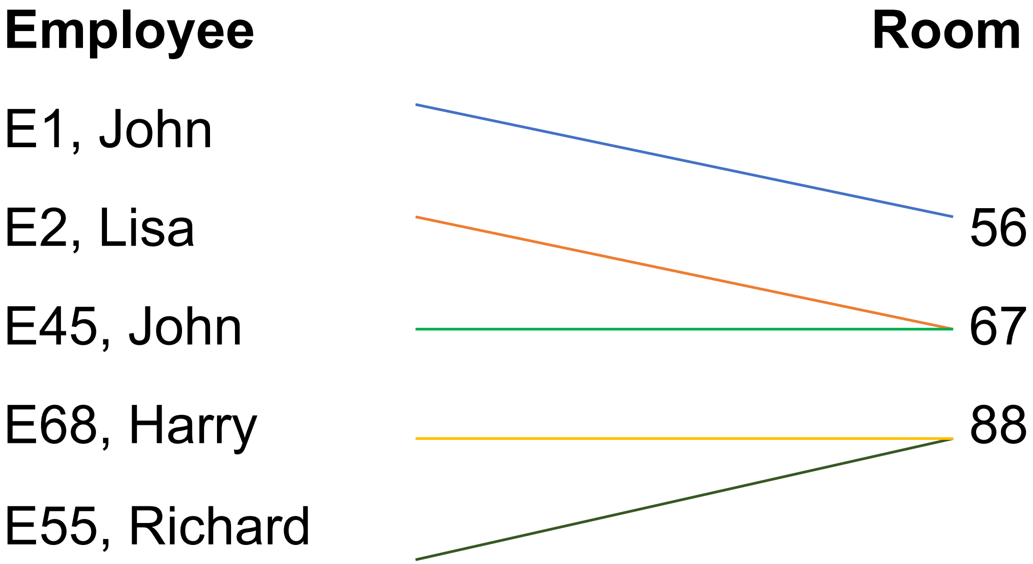
contains a desk of

ROOM

Rno <pi> ROOMNO <M>

Area AREA

Instance level



Employee

E1, John

E2, Lisa

E45, John

E68, Harry

E55, Richard

Room

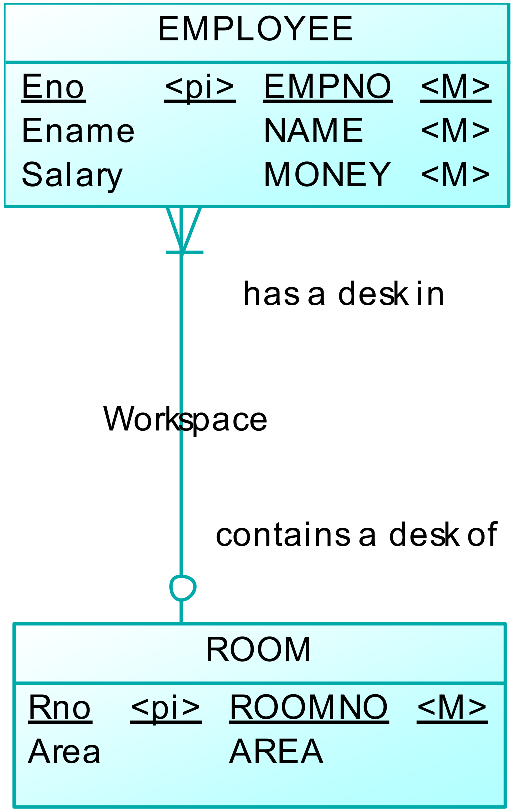
56

67

88

TYPES AND INSTANCES

Type level



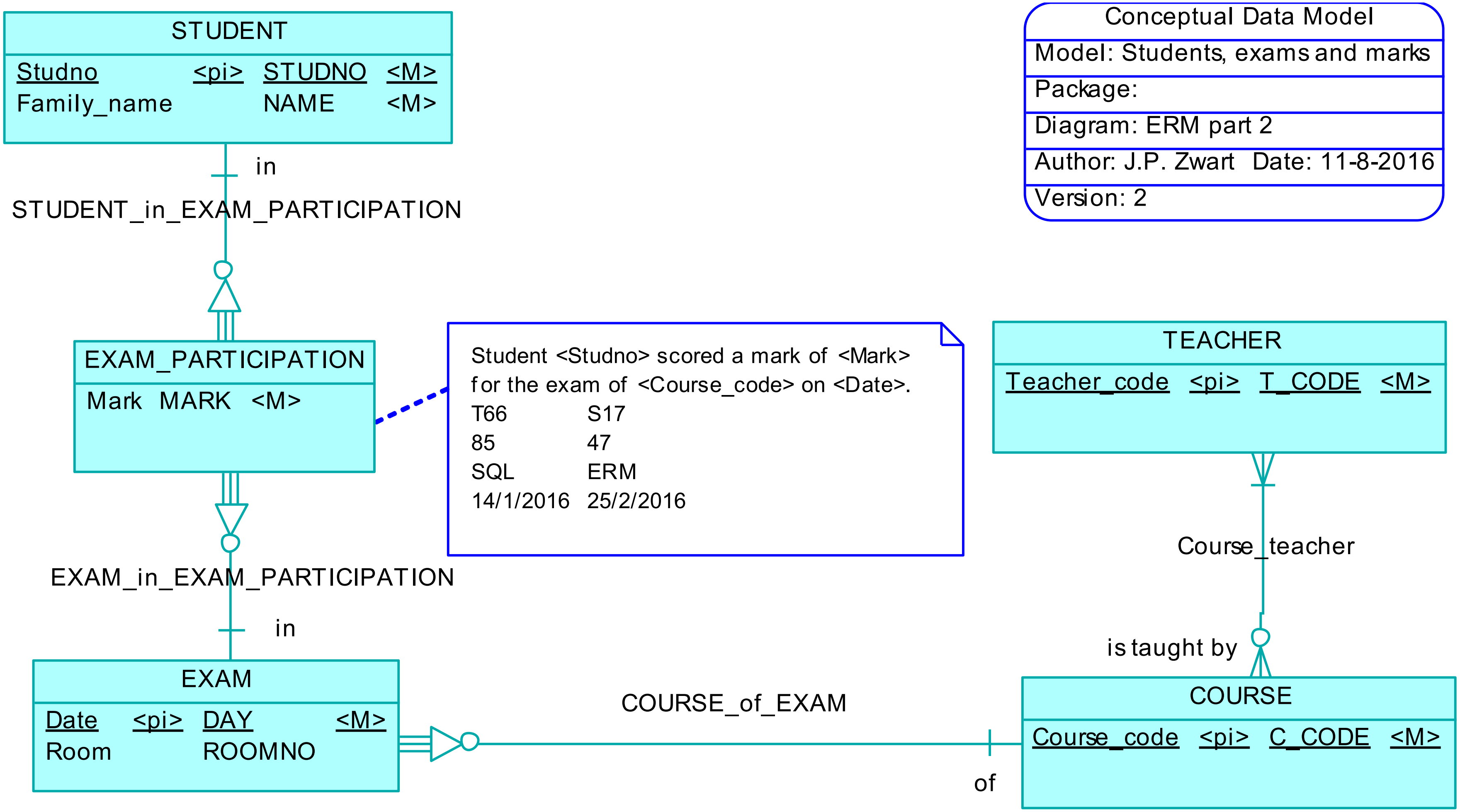
A classic ERM diagram shows only the type level

This suffices for simple everyday ETs and Atts
(but is the meaning of Salary (per month? year?)
and Area (part of building? size?) clear?)

However, for unfamiliar contexts
and/or complex data structures
this is not enough to understand the model.

Adding the **semantics** (meaning) and **examples of instances** to the diagram can greatly help to validate the model (is it correct?).

ANALYZING FACT TYPES: COMPLETE ERD



Conceptual Data Model

Model: Students, exams and marks

Package:

Diagram: ERM part 2

Author: J.P. Zwart Date: 11-8-2016

Version: 2

STUDENT

Studno <pi> STUDNO <M>

Family_name NAME <M>

in

STUDENT_in_EXAM_PARTICIPATION

EXAM_PARTICIPATION

Mark MARK <M>

EXAM_in_EXAM_PARTICIPATION

in

EXAM

Date <pi> DAY <M>

Room ROOMNO

COURSE_of_EXAM

of

Student <Studno> scored a mark of <Mark> for the exam of <Course_code> on <Date>.

T66	S17
85	47
SQL	ERM
14/1/2016	25/2/2016

TEACHER

Teacher_code <pi> T_CODE <M>

Course_teacher

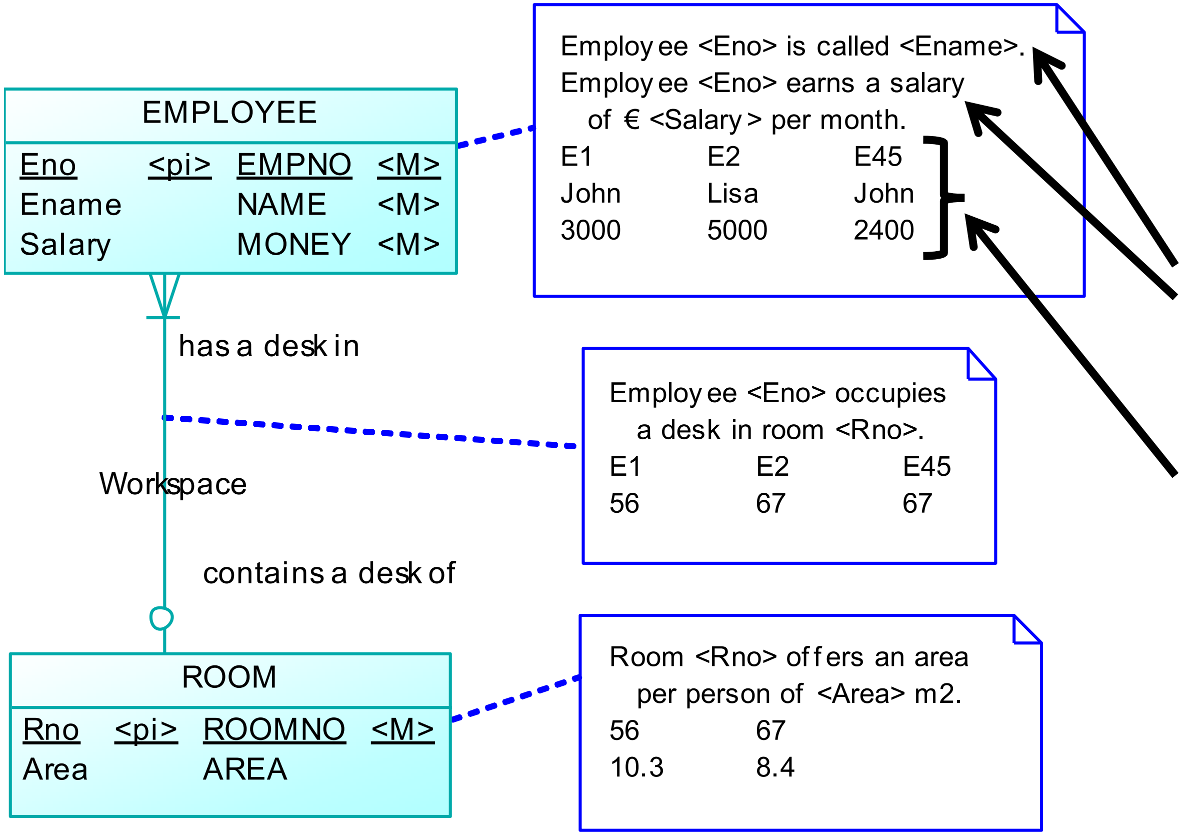
is taught by

COURSE

Course_code <pi> C_CODE <M>

TYPES AND INSTANCES

Type level **Semantics and instance level**



EMPLOYEE

Eno	<pi>	EMPNO	<M>
Ename		NAME	<M>
Salary		MONEY	<M>

ROOM

Rno	<pi>	ROOMNO	<M>
Area		AREA	

Employee <Eno> is called <Ename>.
Employee <Eno> earns a salary of € <Salary> per month.

E1	E2	E45
John	Lisa	John
3000	5000	2400

Employee <Eno> occupies a desk in room <Rno>.

E1	E2	E45
56	67	67

Room <Rno> offers an area per person of <Area> m2.

56	67
10.3	8.4

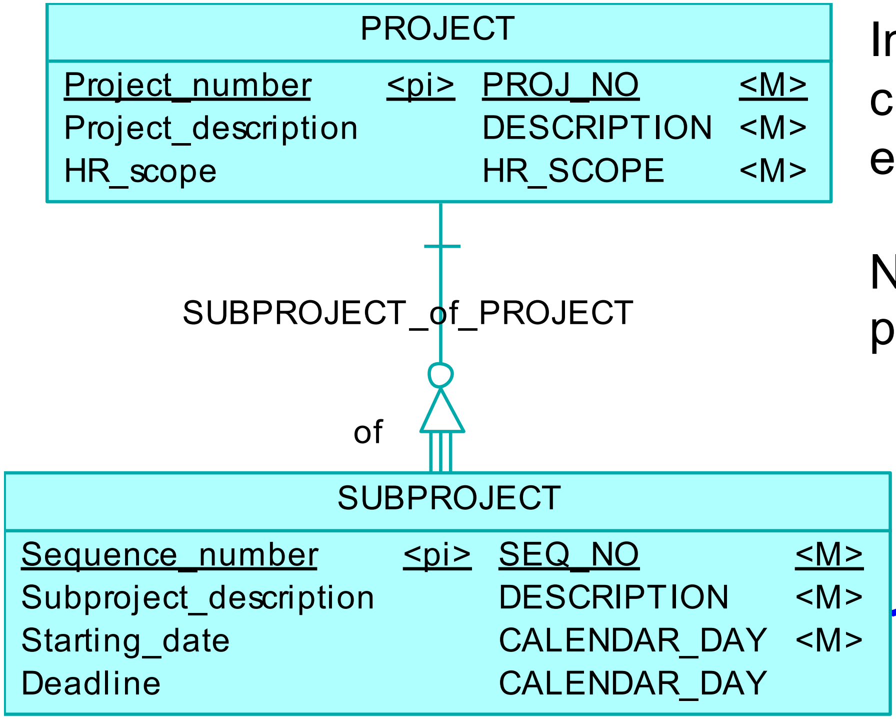
A Fact-Oriented ERM diagram shows predicates and example populations

A predicate represents one type of fact, and makes its meaning clear

The population gives concrete illustrations

In practice:
do this for unclear Atts
and non-dependent RTs

TYPES AND INSTANCES, WEAK ET



PROJECT			
Project_number	<pi>	PROJ_NO	<M>
Project_description		DESCRIPTION	<M>
HR_scope		HR_SCOPE	<M>

SUBPROJECT_of_PROJECT

of

SUBPROJECT			
Sequence_number	<pi>	SEQ_NO	<M>
Subproject_description		DESCRIPTION	<M>
Starting_date		CALENDAR_DAY	<M>
Deadline		CALENDAR_DAY	

Here is a simple example with a weak ET (only one <pi> + Att fact type is shown)

In complex data structures (like branching chains of weak ETs), a predicate and example population can clarify much

Note: a dependent RT cannot have a predicate or population

Subproject <Sequence_number>
of project <Project_number>
must be completed by <Deadline>.

1	2	1
P315	P315	P244
20160205	20160301	20160201

PROCEDURE TO DRAW UP AN ERD

1. Collect concrete examples of facts
   - Use BPM as starting point
   - Make up examples if they don’t exist (yet)
2. Verbalize these examples
   - With domain expert. Result: fact expressions.
   - Make the meaning as clear as possible
3. Sort expressions into Fact Types (FTs)
   - Same kind of expression: same FT
   - Order FTs with most components last
4. Analyze each FT (1 or 2 segments)
   and add the results to the ERD

REMEMBER: RULES FOR ANALYZING FTS

- Mark 2 segments (or 1), and decide on
  ET + Att or ET + ET (if 1 segment: ET).
- If you find an old ET: MATCH
- If you find a new ET:
  determine its ID (primary identifier)
- If this ID contains an ET:
  add a dependent RT to it
- In the ET + ET case:
  add a non-dependent RT
- Give the complete predicate
- Determine <M> for new Atts
- Determine cardinalities for new RTs
- Add predicates and populations to the diagram
  to make the meaning of the fact types more clear

EXAMPLES OF FTS WITH ONE SEGMENT

Example 1: Domain list

```
There is a course ERM.
  "    " "    "   SQL.
```

Such verbalizations might be given for domain lists (departments in an organization, wards in a hospital, towns in a country, …).

Domain lists prevent typos, save users time and effort, and are easily updated by the DB admin.

EXAMPLES OF FTS WITH ONE SEGMENT

Example 1: Domain list

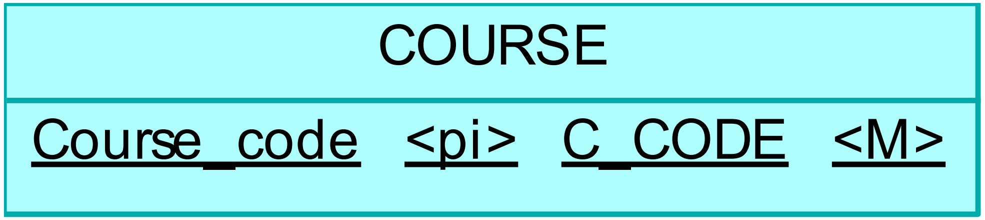
Only one component, only one segment possible.

```
There is a course ERM.
  "    " "    "   SQL.
```

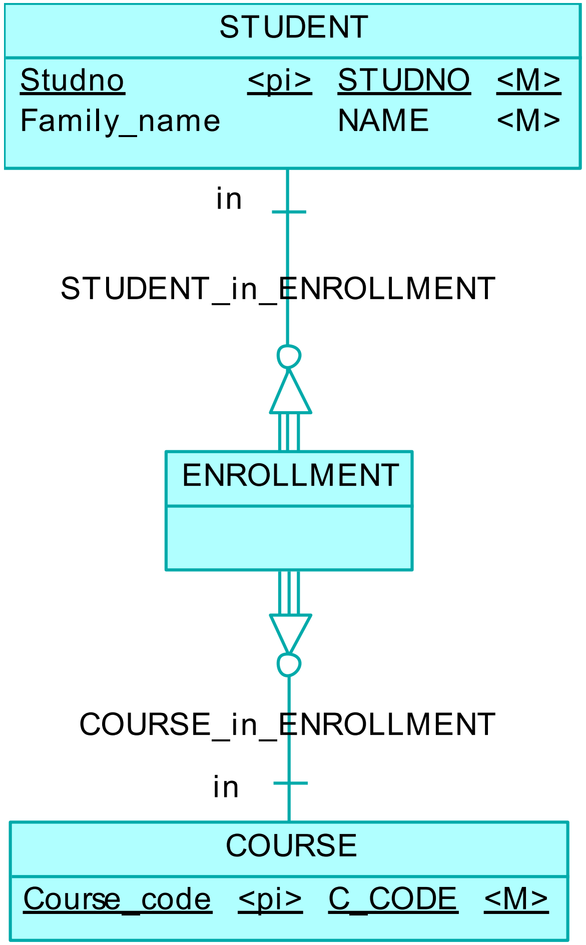
This must then be an ET.

```
There is a course ERM.
  "    " "    “   SQL.
         ET COURSE
         ID: Att Course_code
```

```
Predicate: There is a course
<Course_code>.
```



ANOTHER EXAMPLE WITH A WEAK ET



Note:

- Attributes for ENROLLMENT can be easily added:

```
‘Pending’ is the status of
Att Status                    student S17’s enrollment
                              in the course ERM.

                              ET ENROLLMENT
                              MATCH
```

We have already seen ET ENROLLMENT, so MATCH will do.

- Note that there are no new rules for analyzing FTs with 1 segment. The rules in slide 3 cover all cases.

A COMPLEX EXAMPLE: ANALYSIS

```
FT4:
Student T66 scored a mark of 85 for the exam of SQL on 14/1/2016.
   "    S17   "    "   "    " 47  "   "    "    " ERM  " 25/2/2016.
                   Att Mark

ET EXAM_PARTICIPATION
ID: ET STUDENT + ET EXAM
    MATCH        MATCH
```

Old ETs STUDENT and EXAM present. Mark: attribute (property) of an exam participation.
So other three components must be one ET.

ID contains 2 old ETs: 2 MATCHes

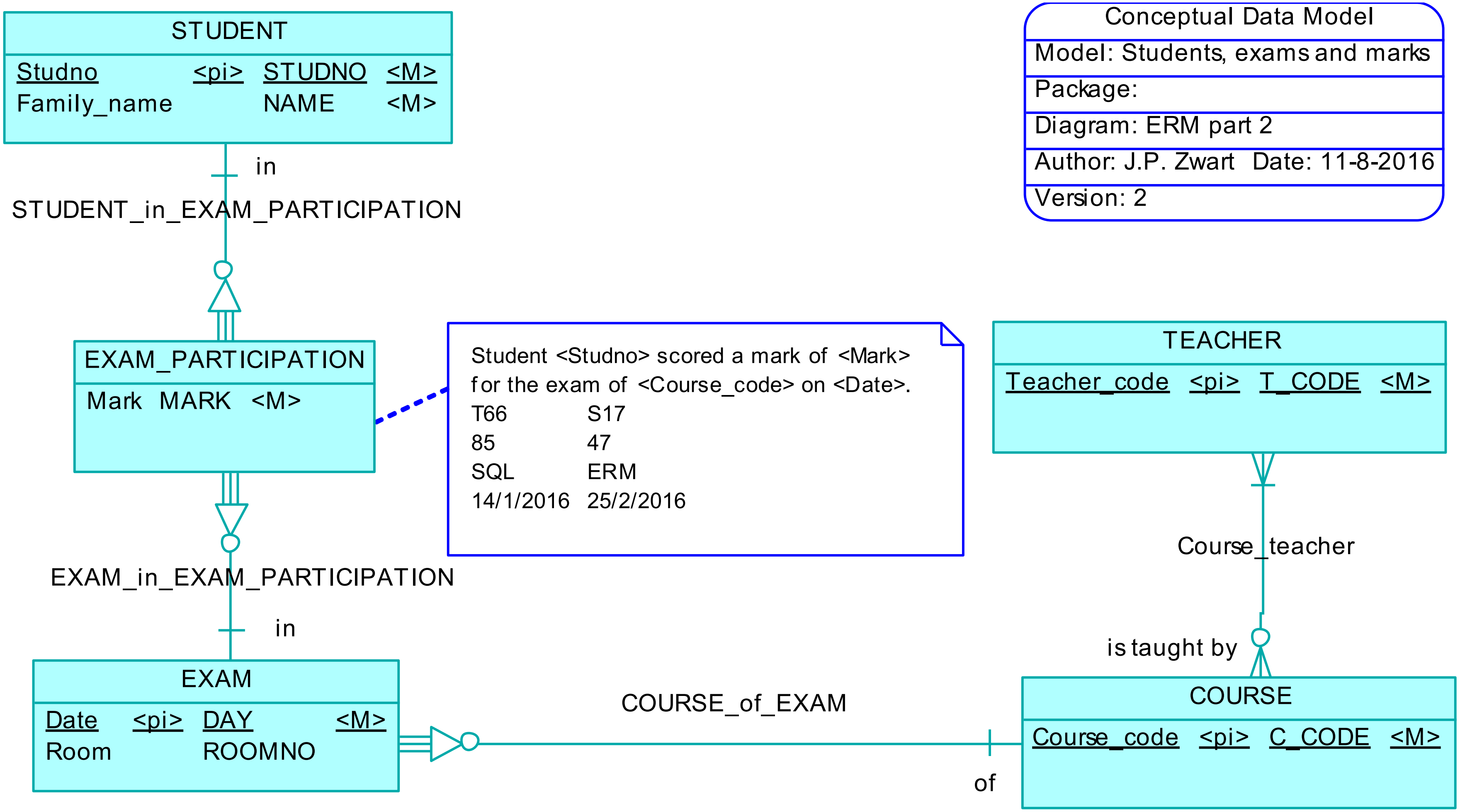
For each ET in the ID: add a dependent RT

```
RT STUDENT_in_EXAM_PARTICIPATION between
   EXAM_PARTICIPATION(dependent) and STUDENT

RT EXAM_in_EXAM_PARTICIPATION between
   EXAM_PARTICIPATION(dependent) and EXAM

Predicate: Student <Studno> scored a mark of <Mark>
           for the exam of <Course_code> on <Date>.
```

A COMPLEX EXAMPLE: THE ERD



STUDENT

Studno <pi> STUDNO <M>
Family_name NAME <M>

in

STUDENT_in_EXAM_PARTICIPATION

EXAM_PARTICIPATION

Mark MARK <M>

EXAM_in_EXAM_PARTICIPATION

in

EXAM

Date <pi> DAY <M>
Room ROOMNO

COURSE_of_EXAM

of

COURSE

Course_code <pi> C_CODE <M>

is taught by

Course_teacher

TEACHER

Teacher_code <pi> T_CODE <M>

Student <Studno> scored a mark of <Mark> for the exam of <Course_code> on <Date>.

T66	S17
85	47
SQL	ERM
14/1/2016	25/2/2016

Conceptual Data Model
Model: Students, exams and marks
Package:
Diagram: ERM part 2
Author: J.P. Zwart Date: 11-8-2016
Version: 2