Database Modeling and Design: The Two-Stage Entity-Relationship Methodology (TSER)

The Two-Stage Entity-Relationship (TSER) Methodology synergistically integrates top-down functional modeling with bottom-up data modeling.

What is TSER?

- Integrated Semantic, Data, and Knowledge modeling methodology.
- Developed by Dr. Cheng Hsu at Rensselaer Polytechnic Institute, NY.
- Focus on functional role of data versus process orientation.
- Two levels (stages) of modeling constructs

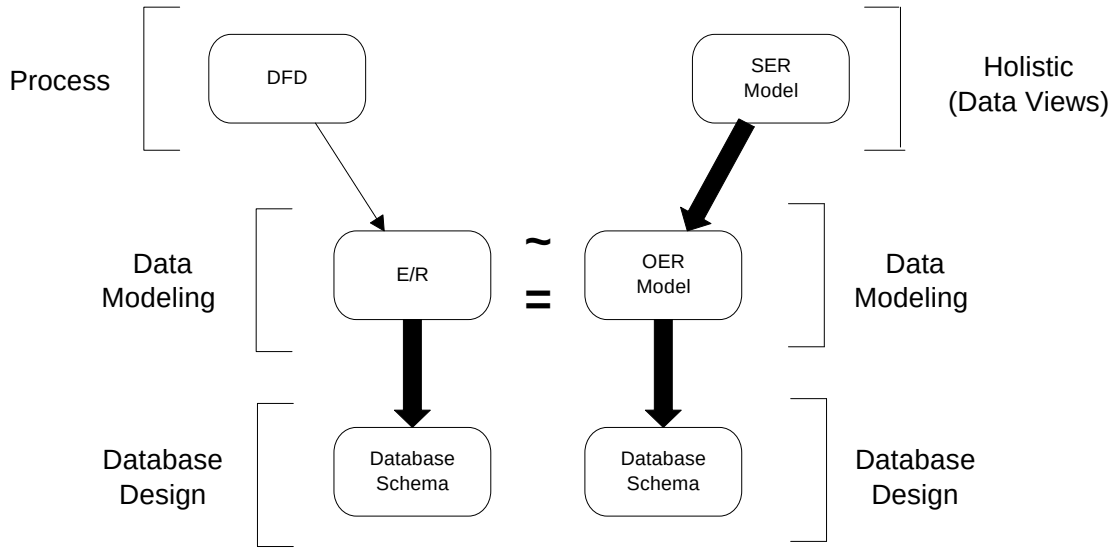
TSER: The Two Stages of Modeling

There are two levels of Entity/Relationship modeling, one at the *semantic* level, the other at the data structure level. The same notion of the classical E/R model has been abstracted to semantic modeling for capturing meanings in the data. Furthermore, the classical E/R which is more data structure oriented has been kept and enhanced as the second level in the TSER methodology and is called the Operational E/R model.

Functional Modeling - Semantic Entity-Relationship (SER) Modeling

Structural Modeling - Operational Entity-Relationship (OER) Modeling

Mapping from SER to OER - Algorithms exist to automate much of the mapping process



The TSER Modeling Constructs

The Modeling Hierarchy

Function Level (SER)

Constructs:

SUBJECT

CONTEXT

Structural Level (OER)

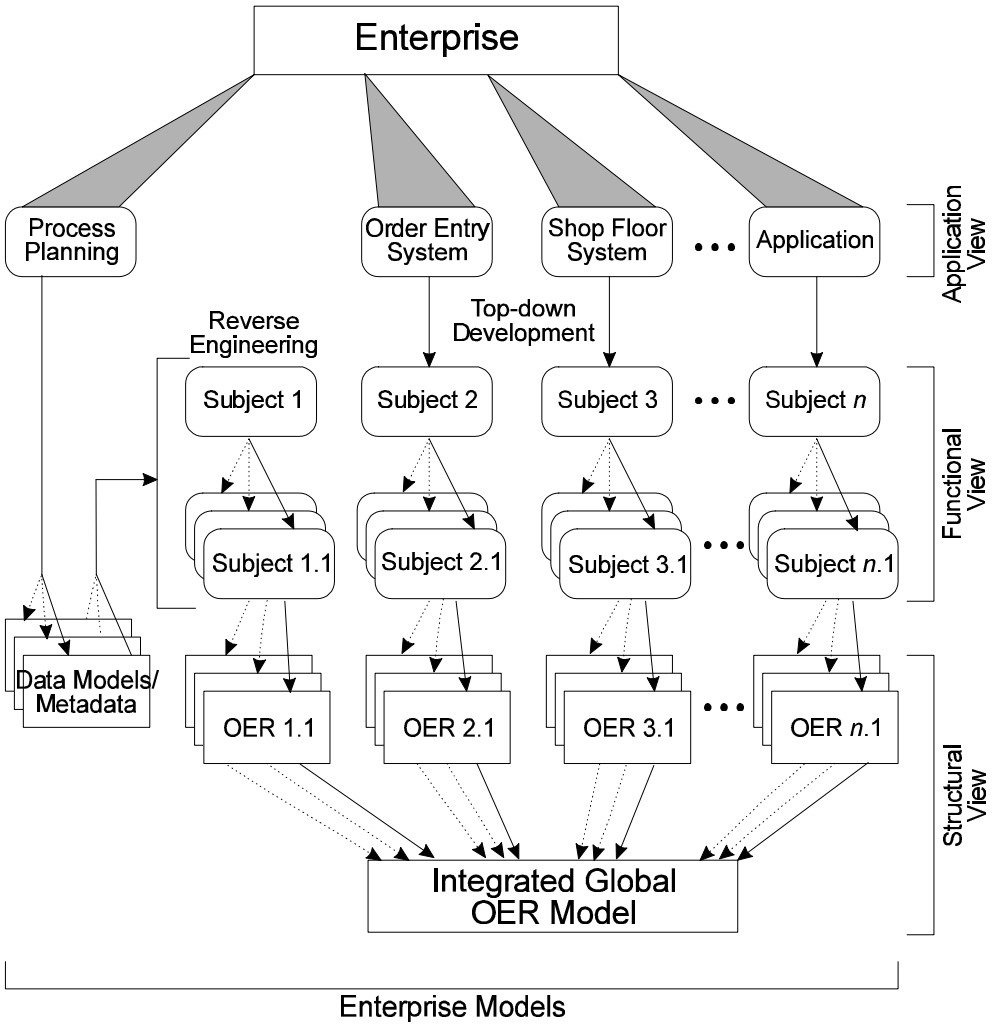
Constructs:

OE (Operational Entity)

PR (Plural Relationship)

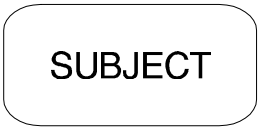
FR (Functional Relationship)

MR (Mandatory Relationship)



The TSER Functional (SER) Modeling Constructs

SUBJECT



- SUBJECTs are a natural unit or grouping of data (e.g., user views, documents, and applications) that both users and analysts can use to describe and characterize the enterprise's information resources and processing.
- The highest-level is named the "application" system (subject) level.
- The level of detail (attributes, rules, inheritance information) specified within the SUBJECT depends entirely on the proposed use of the SER model; as purely a semantic model or targeted to a database design.
- In the case for direct database development, the lowest-level SUBJECT necessitates that its attributes (data items) to be completely specified along with its inter-associations (declared as functional dependencies).
- Internal process logic and business rules can be specified within the SUBJECT as production rules (IF ... THEN ...). (Note: Rules within SUBJECTs are sometimes referred to as intra-subject rules.)

The TSER Functional (SER) Modeling Constructs

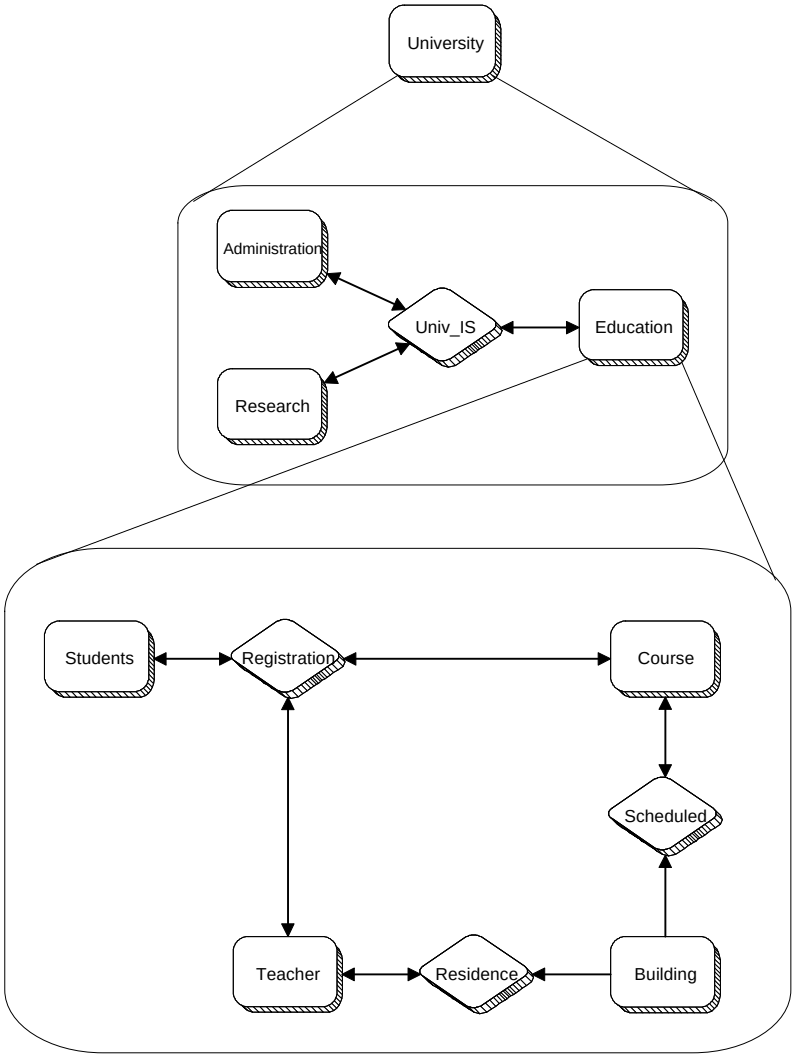
CONTEXT

- CONTEXTs serve to relate two or more SUBJECTs.

- CONTEXTs do not contain data items, but rather inter-subject (production) rules which involve data items from related SUBJECTs. Specifically, the rules represent operating procedures, business controls, data communication, and decision logic can be declared here.

- Arrowed flow lines associated with the CONTEXTs are drawn to represent the direction of information flows and data sharing across SUBJECTs.

Example SER Model:
The University Information System



Details of the University SER Model

Here are some details of the above University SER model. These are representative examples and not all details relating to SUBJECTs/CONTEXTs (e.g., inheritance) are presented.

SUBJECT NAME: University
DESCRIPTION: University Information System
CHILD SUBJECTS: Administration, Research, Education

SUBJECT NAME: Education
DESCRIPTION: Information System for University Education
PARENT SUBJECT: University

Details of the University SER Model (cont.)

SUBJECT NAME: Student
DESCRIPTION: Student Information
PARENT SUBJECT: Education
DATA ITEMS: STUD_NO, STUD_NAME, ADDRESS, DOB, TELEPHONE, ADVISOR, COURSE_NO, CREDITS, TERM, GRADE, CLASS_YR
FDS: STUD_NO -> STUD_NAME, ADDRESS, DOB, TELEPHONE, ADVISOR
(STUD_NO, COURSE_NO) -> TERM, CREDITS, GRADE
RULES:
```
IF
    ((CLASS_YR = 2) AND (Total_Credits(STUD_NO) < 30))
THEN
    Request_Record_Verification(STUD_NO) ;
```
PROC/FNS: Total_Credits, Request_Record_Verification

SUBJECT NAME: Course
DESCRIPTION: Course Information
PARENT SUBJECT: Education
DATA ITEMS: COURSE_NO, COURSE_NAME, PRE-REQ, CREDITS, TERM
FDS: COURSE_NO, TERM -> CREDITS
(COURSE_NO, PRE_REQ) -> (COURSE_NO, PRE_REQ)
COURSE_NO -> COURSE_NAME
ITEM EQUIVALENCE: COURSE_NO = PRE_REQ (In Domain)

SUBJECT NAME: Teacher
DESCRIPTION: Teacher Information
PARENT SUBJECT: Education
DATA ITEMS: STAFF_ID, STAFF_NAME, COURSE_NO, OFFICE, TELEPHONE
FDS: STAFF_ID -> STAFF_NAME, OFFICE, TELEPHONE
RULES: (STAFF_ID, COURSE_NO) -> (STAFF_ID, COURSE_NO)

CONTEXT NAME: Univ_IS
DESCRIPTION: Context relating Administration, Research and Education
SUBJECT LINKS: Administration (↔), Research (↔), Education (↔)

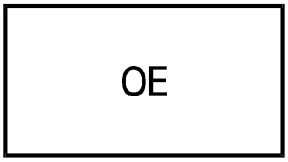
CONTEXT NAME: Registration
DESCRIPTION: Student-Class Registration
SUBJECT LINKS: Student (↔), Course (↔), Teacher (↔)
RULES:
```
IF
    ((Request_Registration(STUD_NO, COURSE_NO) AND
    Course_Taken(STUD_NO, COURSE_NO)) AND
    (Passing_Grade(STUD_NO, COURSE_NO)))
THEN
    Reject_Request(STUD_NO, COURSE_NO) ;
```
PROC/FNS: Request_Registration, Course_Taken, Passing_Grade, Reject_Request

CONTEXT NAME: Scheduled
DESCRIPTION: Student-Class Registration
SUBJECT LINKS: Course (↔), Building (→)
RULES:
```
IF
    ((Request_Registration(STUD_NO, COURSE_NO) AND
    Course_Taken(STUD_NO, COURSE_NO)) AND
    (Passing_Grade(STUD_NO, COURSE_NO)))
THEN
    Reject_Request(STUD_NO, COURSE_NO) ;
```
PROC/FNS: Request_Registration, Course_Taken, Passing_Grade, Reject_Request

... ...

The TSER Structural (OER) Modeling Constructs

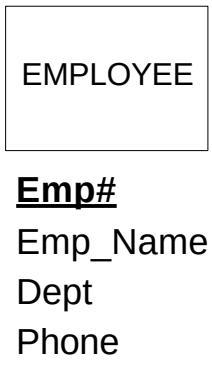
Operational Entity (OE)



A notion concerning the lowest logical units (e.g., a database schema in 3NF) of database system that have unique and independent identity. The OE is always identified by a *singular* primary-key (PK) with all other attributes (including candidate keys) defined as non-key.

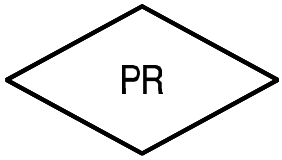
Entity Integrity: States that no component (attribute) of a primary key may accept null value, and no primary key may accept duplicate values.

Example: The following is an Entity representing the notion of Employee. The Primary-Key is in bold and underlined.



The TSER Structural (OER) Modeling Constructs

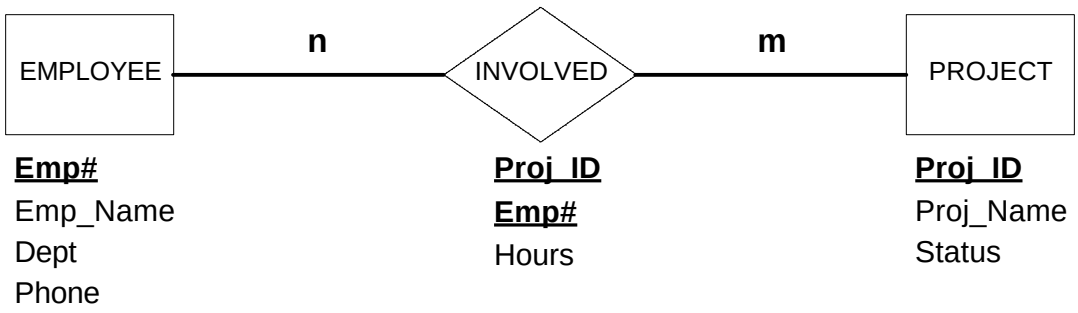
Plural Relationship (PR)



Plural relationship is characterized by many-to-many correspondences of OE's participating in the relationship. A PR is always identified by a *composite* primary-key (PK) that is comprised of the PK's of all OE's involved in the relationship.

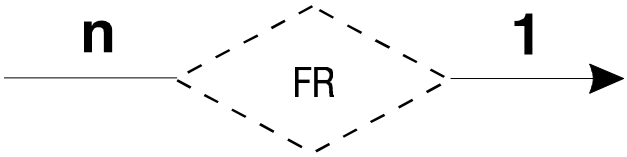
Associative Integrity: States that each of the PR's primary-key attributes itself must be a PK of some OE. Additionally, all the PK's attribute values involved in the PR must match existing ones in the corresponding OE's.

Example: The following is a Plural Relationship representing the idea that a project involves many employees and an employee can be involved in more than one project. Primary-Keys are in bold and underlined.



The TSER Structural (OER) Modeling Constructs

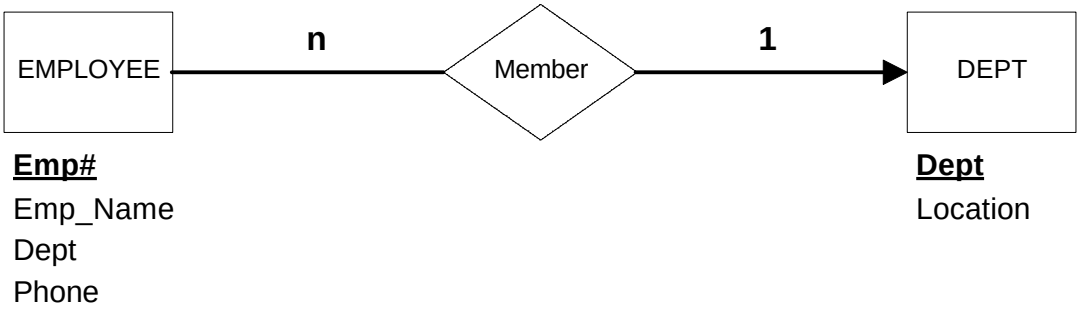
Functional Relationship (FR)



Functional relationship refers to the many-to-one (n:1) correspondences of an attribute between two OEs or an OE/PR pair. An FR implies a *foreign key* relationship and hence the existence of a referential integrity. The FR is a binary relationship with an "arrow" side representing the *determined* (or dependent) side and points to an OE, while the other side is called the *determinant* and is also linked to either an OE or a PR. The primary key of the determined side is included as a non-prime attribute (i.e., a foreign key) of the determinant side.

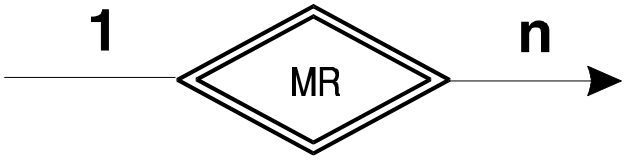
Referential Integrity: States that the value of every foreign key in the *determinant* must either be null or be identical to some PK value in the *determined*.

Example: The following is a Functional Relationship representing the concept that a department can have many employees, but an employee can belong to only one department. Keys are in bold and underlined.



The TSER Structural (OER) Modeling Constructs

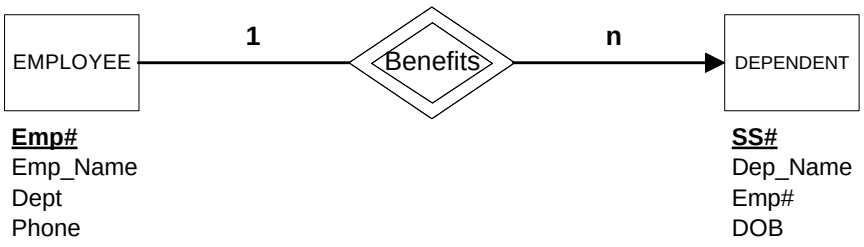
Mandatory Relationship (MR)



Mandatory relationship refers to the one-to-many correspondences between OEs that have an existence dependency defined between them - this is termed a "fixed" association. As a binary relationship, the arrow side points to the *owned* OE (e.g., weak entity) while the other side is linked to the *owner* OE. There is a foreign key implied in the owned OE whose value must match exactly the owner's PK value.

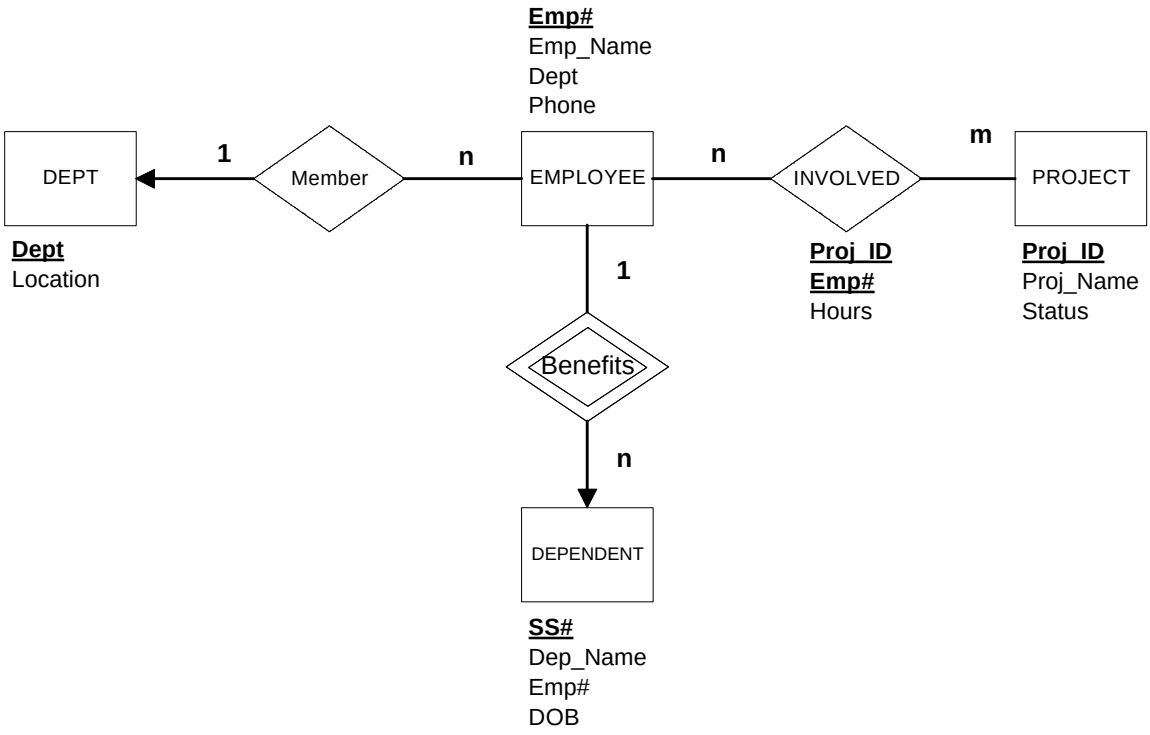
Existence Dependency: States that when the *owner* entity instance is deleted, then all *owned* entity instances associated will it must also be deleted, if and only if, the owned OEs do not participate in a PR, FR nor as the owned of another MR. (This is similar to the notion of the "weak entity" in the original Entity-Relationship approach by P. Chen with the difference being the permissible retention of owned OE's.)

Example: The following is a Mandatory Relationship representing the idea that an employee can have many dependents and the dependents existence (in a database) is directly based on the existence of the employee. Keys are in bold and underlined.



Example OER Model

The following OER model represents the assembly of the above OER model components of the Employee-Department-Project-Dependent model examples above.



Data Modeling Through the TSER Approach

The overall procedure of TSER modeling can be summarized as follows:

(1) Conduct a high-level systems analysis (i.e., top-down modeling) using the SER modeling constructs. Fill in all details (e.g., data items, FDs, rules) necessary for the SER modeling hierarchy.

(2) Map the analysis into a SER model.

(3) Map the SER model into the normalized OER model. (Note: The IBMS CASE Tool automates this process.)

(4) Map the OER to the required database schema (SQL, dBase, MS-Access, OODB, etc.)

Note: It is possible to use other high-level modeling methodologies (DFD, IDEF0, OO, etc.) then identify a mapping from that representation to the SER model. This paradigm mapping capability to SER is possible due to the generic nature of the TSER model.

In sum, underlying these constructs at both stages are two types of primitives: data items and predicate logic. Therefore, the basic structure of TSER metadata is characterized by (1) data representation as relations, (2) knowledge representation in the form of production rules, and (3) the two representations being tied via data items.

Mapping from SER to OER

- Algorithms are also developed for mapping the SER model to a normalized (at least third-normal form) OER model [Hsu 1985; Hsu et al. 1993].

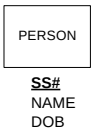
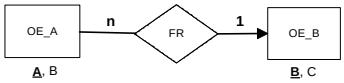
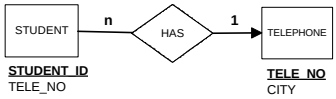
- The OER model itself can be directly mapped to a (relational, network, and object-oriented) database implementation.

- The companion CASE Tool, Information-Based Modeling System (IBMS), supports the TSER methodology in its entirety.

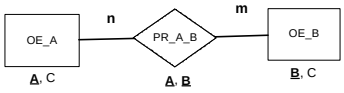
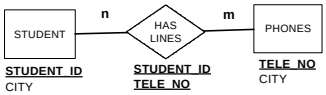
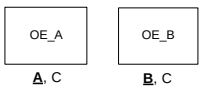
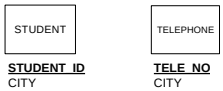
- The IBMS CASE Tool automates the mapping process from SER to OER and from OER to a database schema. The mapping requires some user intervention in the process for refinement of the OER model, in terms of integrity constraint definitions.

Functional Dependency in TSER Mapping Examples

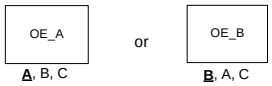
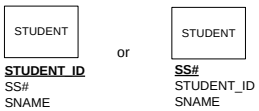
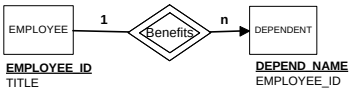
The following examples for Functional Dependency declaration is demonstrated with a view towards good database design. As such, a walkthrough of the mapping process of the SER to OER to database schema (SQL with no integrity constraint declaration) mapping routines for FDs are given for each example.

	Conceptual Example	Detailed Example
Example 1 Subject1 FD Set:	$A \rightarrow B, C$	$SS\# \rightarrow NAME, DOB$
OER:		
SQL:	`CREATE TABLE OE_A` `(A CHAR(10) NOT NULL,` `B CHAR(10)` `C CHAR(10));`	`CREATE TABLE PERSON` `(SS# CHAR(10) NOT NULL,` `NAME CHAR(40)` `DOB CHAR(10));`
Example 2 Subject1 FD Set:	$A \rightarrow B$ $B \rightarrow C$	$STUDENT\_ID \rightarrow TELE\_NO$ $TELE\_NO \rightarrow CITY$
OER:		
SQL:	`CREATE TABLE OE_A` `(A CHAR(10) NOT NULL,` `B CHAR(10));` `CREATE TABLE OE_B` `(B CHAR(10) NOT NULL,` `C CHAR(10));`	`CREATE TABLE STUDENT` `(STUDENT_ID CHAR(10) NOT NULL,` `TELE_NO CHAR(10));` `CREATE TABLE TELEPHONE` `(TELE_NO CHAR(10) NOT NULL,` `CITY CHAR(40));`

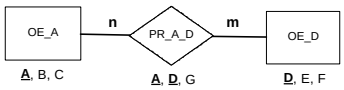
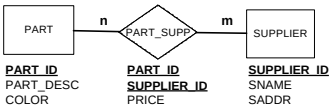
Functional Dependency in TSER Mapping Examples

	Conceptual Example	Detailed Example
Example 3A Subject1 FD Set:	$A \rightarrow C$ $B \rightarrow C$	STUDENT_ID → CITY TELE_NO → CITY
OER:		
SQL:	`CREATE TABLE OE_A` `(A CHAR(10) NOT NULL,` `C CHAR(10));` `CREATE TABLE OE_B` `(B CHAR(10) NOT NULL,` `C CHAR(10));` `CREATE TABLE PR_A_B` `(A CHAR(10) NOT NULL,` `B CHAR(10) NOT NULL);`	`CREATE TABLE STUDENT` `(STUDENT_ID CHAR(10) NOT NULL,` `CITY CHAR(40));` `CREATE TABLE TELEPHONE` `(TELE_NO CHAR(10) NOT NULL,` `CITY CHAR(40));` `CREATE TABLE HAS_LINES` `(STUDENT_ID CHAR(10) NOT NULL,` `TELE_NO CHAR(10) NOT NULL);`
Example 3B Subject1 FD Set: Subject2 FD Set:	$A \rightarrow C$ $B \rightarrow C$	STUDENT_ID → CITY TELE_NO → CITY
OER:		
SQL:	`CREATE TABLE OE_A` `(A CHAR(10) NOT NULL,` `C CHAR(10));` `CREATE TABLE OE_B` `(B CHAR(10) NOT NULL,` `C CHAR(10));`	`CREATE TABLE STUDENT` `(STUDENT_ID CHAR(10) NOT NULL,` `CITY CHAR(40));` `CREATE TABLE TELEPHONE` `(TELE_NO CHAR(10) NOT NULL,` `CITY CHAR(40));`

Functional Dependency in TSER Mapping Examples

	Conceptual Example	Detailed Example
Example 4 Subject1 FD Set:	A → C A ↔ B	STUDENT_ID → SNAME STUDENT_ID ↔SS#
OER:		
SQL:	`CREATE TABLE OE_A` `  (A CHAR(10) NOT NULL,` `  B CHAR(10),` `  C CHAR(10));` ***or*** *`CREATE TABLE OE_B`* *`  (B CHAR(10) NOT NULL,`* *`  A CHAR(10),`* *`  C CHAR(10));`*	`CREATE TABLE STUDENT` `  (STUDENT_ID CHAR(10) NOT NULL,` `  SS# CHAR(10),` `  SNAME CHAR(40));` ***or*** *`CREATE TABLE STUDENT`* *`  (SS# CHAR(10) NOT NULL,`* *`  STUDENT_ID CHAR(10),`* *`  SNAME CHAR(40));`*
Example 5 Subject1 FD Set:	A → B C <← A	EMPLOYEE_ID → TITLE DEPEND_NAME <← EMPLOYEE_ID
OER:		
SQL:	`CREATE TABLE OE_C` `  (C CHAR(10) NOT NULL,` `  A CHAR(10));` `CREATE TABLE OE_A` `  (A CHAR(10) NOT NULL,` `  B CHAR(10));`	`CREATE TABLE EMPLOYEE` `  (EMPLOYEE_ID CHAR(10) NOT NULL,` `  TITLE CHAR(30));` `CREATE TABLE DEPENDENT` `  (DEPEND_NAME CHAR(10) NOT NULL,` `  EMPLOYEE_ID CHAR(10));`

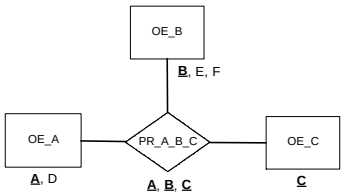
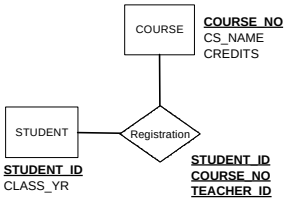
Functional Dependency in TSER Mapping Examples

<table>
<tr><th></th><th>Conceptual Example</th><th>Detailed Example</th></tr>
<tr><td>Example 6
Subject1
FD Set:</td><td>A → B, C
D → E, F
(A, D) → G</td><td>PART_ID → PART_DESC, COLOR
SUPPLIER_ID → SNAME, SADDR
(PART_ID, SUPPLIER) → PRICE</td></tr>
<tr><td>OER:</td><td> 
</td><td> 
</td></tr>
<tr><td>SQL:</td><td><pre>CREATE TABLE OE_A
    (A CHAR(10) NOT NULL,
     B CHAR(10),
     C CHAR(10));

CREATE TABLE OE_D
    (D CHAR(10) NOT NULL,
     E CHAR(10),
     F CHAR(10));

CREATE TABLE PR_A_D
    (A CHAR(10) NOT NULL,
     D CHAR(10) NOT NULL,
     G CHAR(10));</pre></td><td><pre>CREATE TABLE PART
    (PART_ID CHAR(10) NOT NULL,
     PART_DESC CHAR(40),
     COLOR CHAR(10));

CREATE TABLE SUPPLIER
    (SUPPLIER_ID CHAR(10) NOT NULL,
     SNAME CHAR(30),
     SADDR CHAR(80));

CREATE TABLE PART_SUPP
    (PART_ID CHAR(10) NOT NULL,
     SUPPLIER_ID CHAR(10) NOT NULL,
     PRICE NUMBER);</pre></td></tr>
<tr><td>Example 7
Subject1
FD Set:</td><td>(A, B, C) → (A, B, C)
A → D
B → E, F</td><td>STUDENT_ID, COURSE_NO, TEACHER_ID →
STUDENT_ID, COURSE_NO, TEACHER_ID
STUDENT_ID → CLASS_YR
COURSE_NO → CS_NAME, CREDITS</td></tr>
<tr><td>OER:</td><td> 
</td><td> 
</td></tr>
<tr><td>SQL:</td><td><pre>CREATE TABLE OE_A
    (A CHAR(10) NOT NULL,
     D CHAR(10));

CREATE TABLE OE_B
    (B CHAR(10) NOT NULL,
     E CHAR(10),
     F CHAR(10));

optional:
CREATE TABLE OE_C
    (C CHAR(10) NOT NULL);

CREATE TABLE PR_A_B_C
    (A CHAR(10) NOT NULL,
     B CHAR(10) NOT NULL,
     C CHAR(10) NOT NULL);</pre></td><td><pre>CREATE TABLE STUDENT
    (STUDENT_ID CHAR(10) NOT NULL,
     CLASS_YR CHAR(2));

CREATE TABLE COURSE
    (COURSE_NO CHAR(10) NOT NULL,
     CS_NAME CHAR(30),
     CREDITS NUMBER);

optional:
CREATE TABLE TEACHER
    (TEACHER_ID CHAR(10) NOT NULL);

CREATE TABLE REGISTRATION
    (STUDENT_ID CHAR(10) NOT NULL,
     COURSE_NO CHAR(10) NOT NULL,
     TEACHER_ID CHAR(10) NOT NULL);</pre></td></tr>
</table>

Database Design - OER to Database Schema

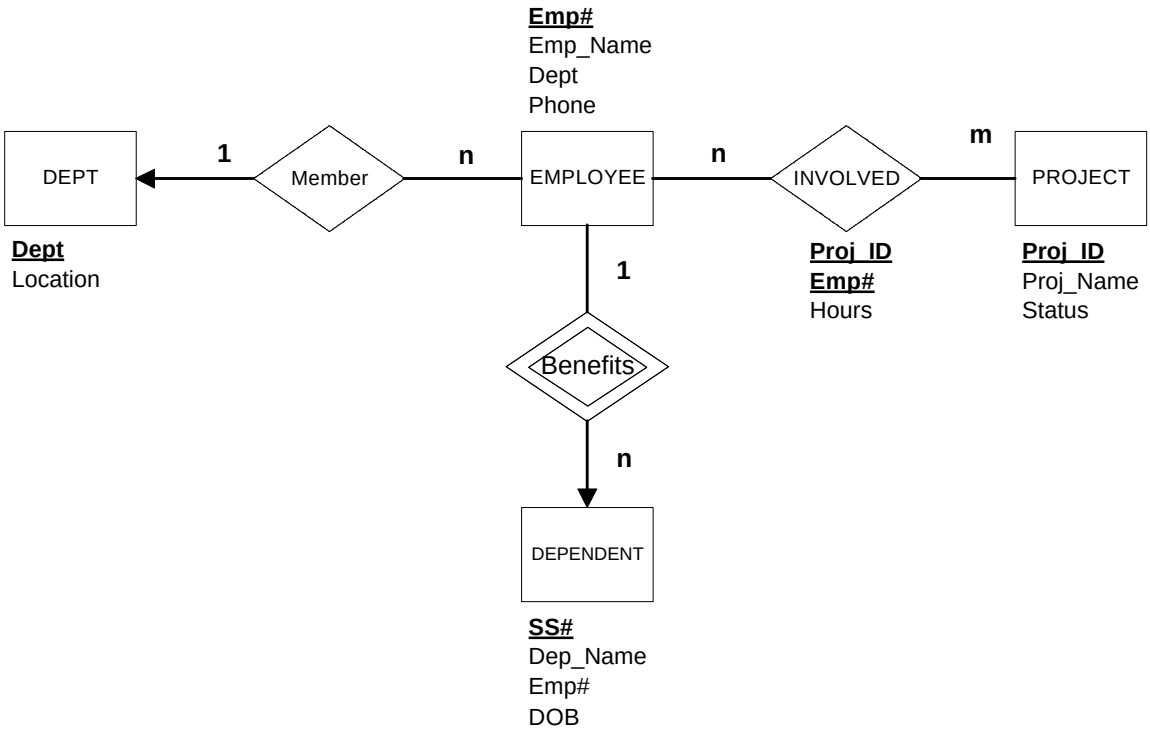
Please note, in the following examples, field-definitions are not all declared. Place-holders (...) are presented for the sake of clarity. SQL/92 data definition language used for reference and includes the integrity constraint definitions. No index is defined for the relations.

The following example is the classic Employee-Department-Project-Dependent E/R model found in database textbooks. The relationships in the following OER model representation states:

(1) A department can have many employees, but an employee can belong to only one department.

(2) A project has many employees, and an employee can be involved in more than one project.

(3) An employee can have many dependents, but the dependents existence is directly based on the existence of the employee.



OER to Database Schema (SQL/92)

Operational Entity (OE) -> SQL/92

```
   CREATE BASE RELATION DEPT
          ( DEPT      ... ,
                      ... ,
                      ... )
            PRIMARY KEY ( DEPT ) ;

   CREATE BASE RELATION EMPLOYEE
          ( EMP#      ... ,
            DEPT      ... ,
                      ... )
            PRIMARY KEY ( EMP# ) ;

   CREATE BASE RELATION DEPENDENT
          ( SS# ... ,
            EMP#      ... ,
                      ... )
            PRIMARY KEY ( SS# ) ;

   CREATE BASE RELATION PROJECT
          ( PROJ_ID   ... ,
                      ... ,
                      ... )
            PRIMARY KEY ( PROJ_ID ) ;
```

Plural Relationship (PR) -> SQL/92

```
 CREATE BASE RELATION EMP-PROJ
    ( EMP#      ... ,
      PROJ_ID   ... ,
                ... )
      PRIMARY KEY ( EMP#, PROJ_ID )
      FOREIGN KEY ( EMP# ) REFERENCES EMPLOYEE
      FOREIGN KEY ( PROJ_ID ) REFERENCES PROJECT
 ;
```

OER to Database Schema (SQL/92)

Functional Relationship (FR) -> SQL/92

No relations are created, but rather the following modifications are made to the SQL declaration of the related OE EMPLOYEE.

```
CREATE BASE RELATION EMPLOYEE
   ( EMP#       ... ,
     DEPT       ... ,
                ... )
     PRIMARY KEY ( EMP# )
     FOREIGN KEY ( DEPT ) REFERENCES DEPT ;
```

Mandatory Relationship (MR) -> SQL/92

No relations are created, but rather the following modifications are made to the SQL declaration of the related OE DEPENDENT.

```
CREATE BASE RELATION DEPENDENT
   ( SS#        ... ,
     EMP#       ... ,
                ... )
     PRIMARY KEY ( SS# )
     FOREIGN KEY ( EMP# ) REFERENCES EMPLOYEE
                 DELETE CASCADES
                 UPDATE CASCADES ;
```

The IBMS CASE Tool

Definition

IBMS = Information Based Modeling System

General Information

Implements the TSER Modeling Methodology.

University research product released as educational freeware.

Purpose

Semantic Modeler.

Database Design Tool.

History

Began as a LISP language-based system on a Xerox Workstation.

Moved to a PC-DOS based LISP system.

Migrated to a Unix-based implementation in X-Windows Motif.

Latest incarnation is an MS-Windows based system.

Where to get?

World Wide Web Source: http://viu.eng.rpi.edu

DDA Homepage: http://www.cs.hku.hk/~cs272

Where to get the IBMS CASE Tool in HKU?

HKUCSD3 Novell Server

Executable from Novell Server

f:\apps\ibms*.exe

Compressed Files for Download

f:\apps\ibms\download\tser1.exe
(or ibms1.zip)

f:\apps\ibms\download\tser2.exe
(or ibms2.zip)

Documentation Set in Adobe Acrobat v3

f:\apps\ibms\docs\ibms_man.pdf

f:\apps\ibms\docs\concept.pdf

Diskette Preparation

Diskette 1: tser1.exe (or ibms1.zip)

Diskette 2: tser2.exe (or ibms2.zip)

Diskette 3: ibms_man.pdf
concept.pdf