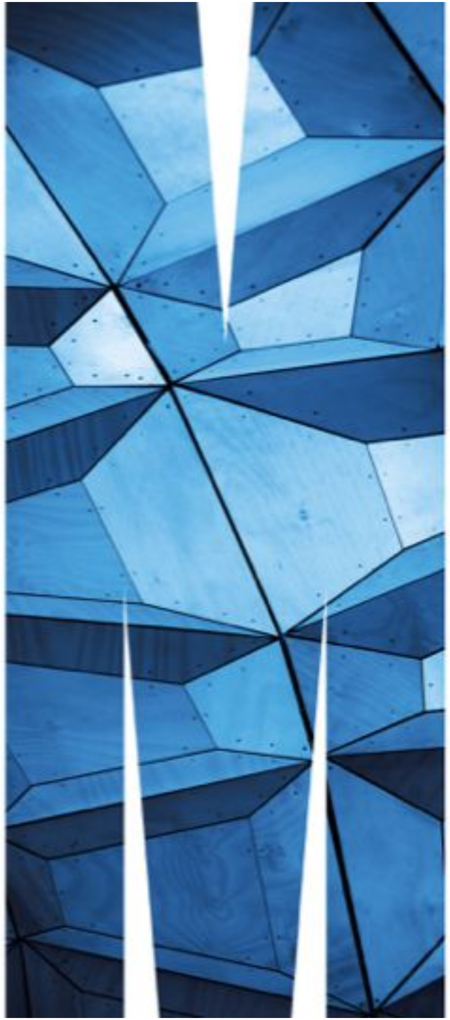
MONASH
INFORMATION
TECHNOLOGY

Topic 5
Database Design II: Logical Modelling

2025 S1



Reference

Several of the examples and diagrams used this week have been taken from:

Hoffer, J. A. , Prescott, M. B. & McFadden, F. R. “Modern Database Management”

Step 2 (and 3) of the Design Process

- Step 1 Conceptual Model (Topic 2)
  - Database Model independent
- Step 2 Logical Model (this topic)
  - Select which type (model) of database you wish to implement your conceptual model in
    - Network, Relational, OO, XML, NoSQL, ...
  - Database model dependent
- Step 3 Physical Model
  - Select which specific vendor for your chosen model you will implement in
    - Oracle, MySQL, IBM DB2, SQL Server, ...
  - Database vendor dependent
  - Final output schema file to implement model (for relational model a set of tables)

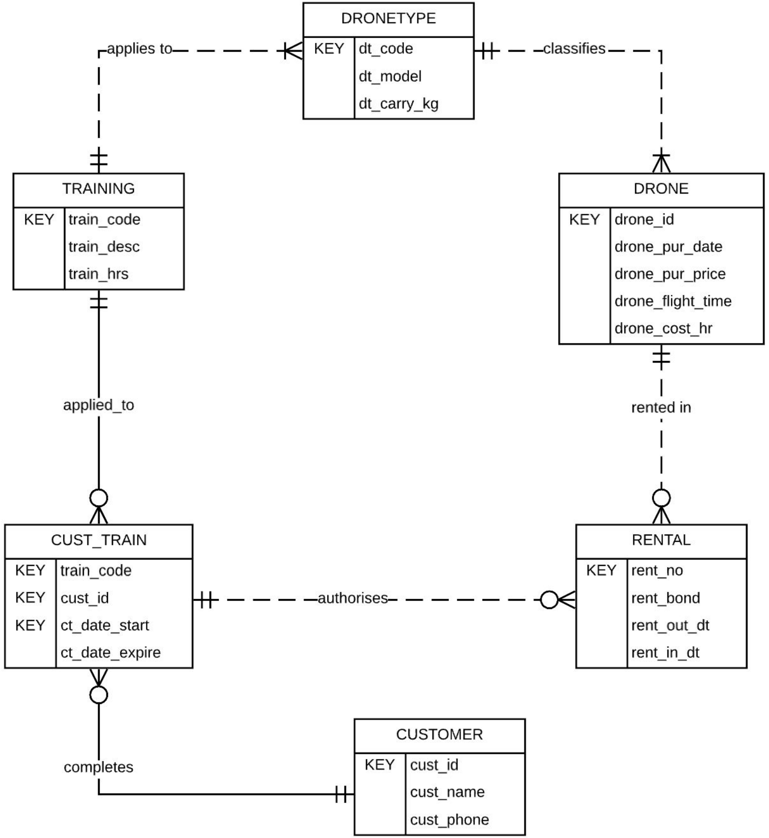
Summary of Terminologies at Different Levels

Conceptual	Logical (Relational)	Physical (Relational)
Entity	Relation	Table
Attribute	Attribute	Column
Instance	Tuple	Row
Identifier (KEY)	Primary Key	Primary Key
Relationship	---	---
---	Foreign Key	Foreign Key

Topic 3 Relational Model Characteristics

- Each relation must have a unique name
- Each attribute of a relation must have a distinct name within the relation
- An attribute cannot be multivalued (consist of repeating values)
- All values of an attribute need to be from the same domain
- The order of attributes and tuples in a relation is immaterial
- Each relation must have a primary key
- Logical (not physical) connections are made between relations by virtue of primary/foreign key pairing

Recap Topic 2 HiFlying Drones Conceptual Model



DRONETYPE	
KEY	dt_code
	dt_model
	dt_carry_kg

TRAINING	
KEY	train_code
	train_desc
	train_hrs

DRONE	
KEY	drone_id
	drone_pur_date
	drone_pur_price
	drone_flight_time
	drone_cost_hr

CUST_TRAIN	
KEY	train_code
KEY	cust_id
KEY	ct_date_start
	ct_date_expire

RENTAL	
KEY	rent_no
	rent_bond
	rent_out_dt
	rent_in_dt

CUSTOMER	
KEY	cust_id
	cust_name
	cust_phone

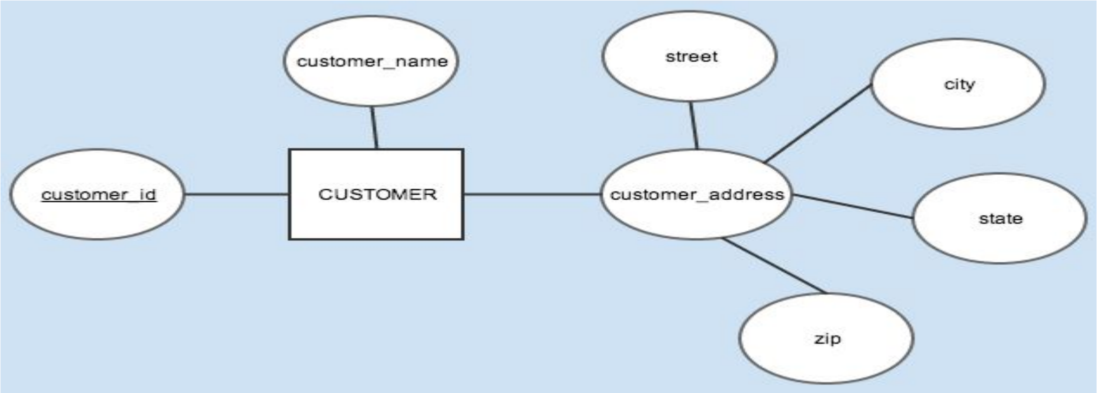
Transforming ER diagrams into relations (mapping conceptual level to logical level)

- Essentially
  - KEY to PK
  - Represent relationships with PK/FK pairs
- The steps are:
  - Map strong (regular) entities
  - Map weak entities
  - Map binary relationships
  - Map associative entities
  - Map unary relationships
  - Map ternary relationships
  - *Map supertype/subtype relationships (is not part of this unit).*

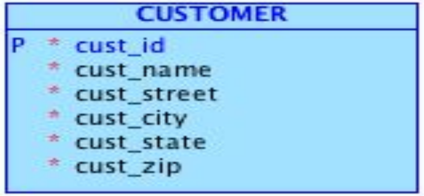
Map Regular Entities

- Composite Attributes
  - When the regular entity type contains a composite attribute, only the simple component attributes of the composite attribute are included in the new relation.
  - Compared to composite attributes, simple attributes not only improve data accessibility but also help in maintaining data quality
  - *Mapping a composite to its simple component attributes is the normal action* ***if no client specification, to the contrary, is available***
    - however, if in doubt ask eg phone numbers

Mapping a Composite Attribute



CUSTOMER	
Key	cust_id
	cust_name
	cust_street
	cust_city
	cust_state
	cust_zip



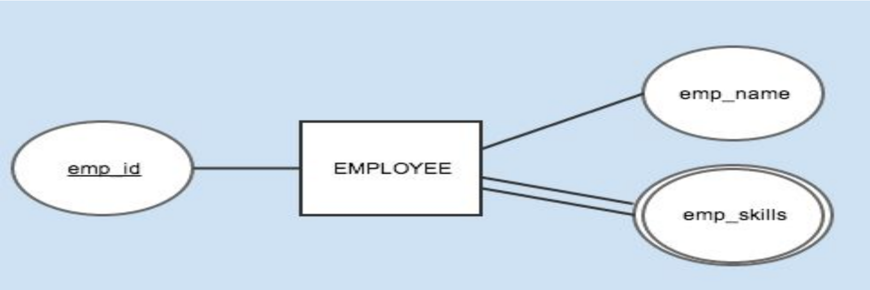
CUSTOMER

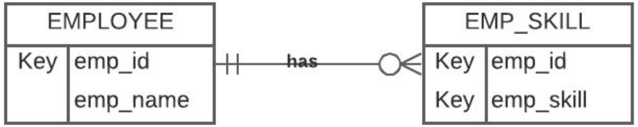
```
P * cust_id
  * cust_name
  * cust_street
  * cust_city
  * cust_state
  * cust_zip
```

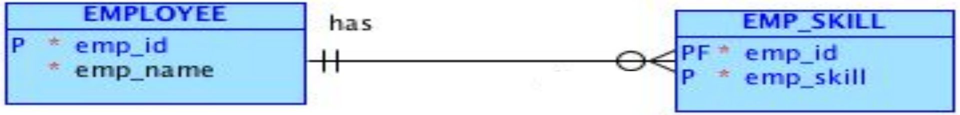
Map Regular Entities

- Multivalued Attribute
  - When the regular entity type contains a multivalued attribute, two new relations are created.
  - The first relation contains all the attributes of the entity type except the multivalued attribute itself.
  - The second relation contains two attributes that form the PK. One of the attributes is the PK from the first relation, which becomes the FK in the second relation and the other is the multivalued attribute.
  - There can also be non key attributes in the second relation depending upon the data requirements.

Mapping a Multi valued Attribute





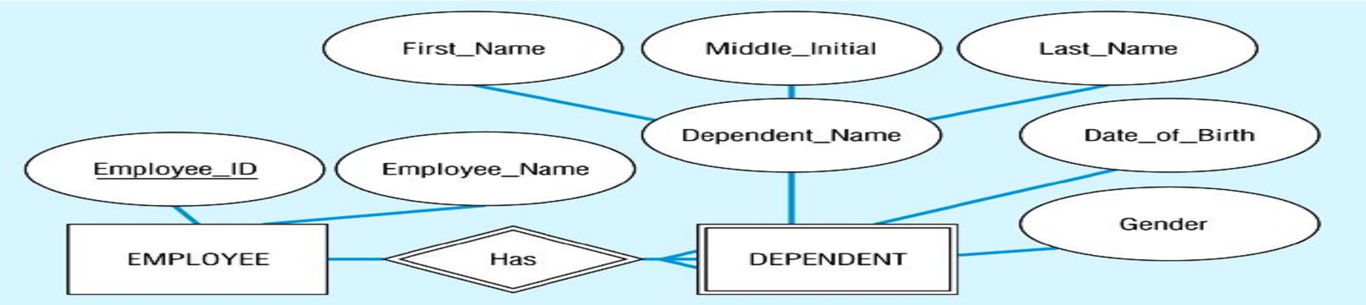


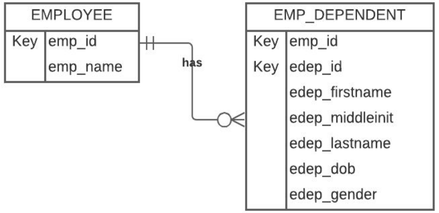
Is there a better solution than the one shown above?

What are the issues here - this was partially discussed in Topic 2

Mapping a Weak Entity

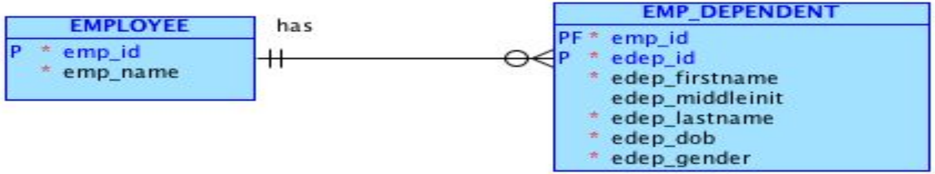
- For each weak entity type, create a new relation and include all of the simple attributes as attributes of this relation. The PK of the identifying relation is also included as the FK in this new relation.





EMPLOYEE	
Key	emp_id
	emp_name

EMP_DEPENDENT	
Key	emp_id
Key	edep_id
	edep_firstname
	edep_middleinit
	edep_lastname
	edep_dob
	edep_gender

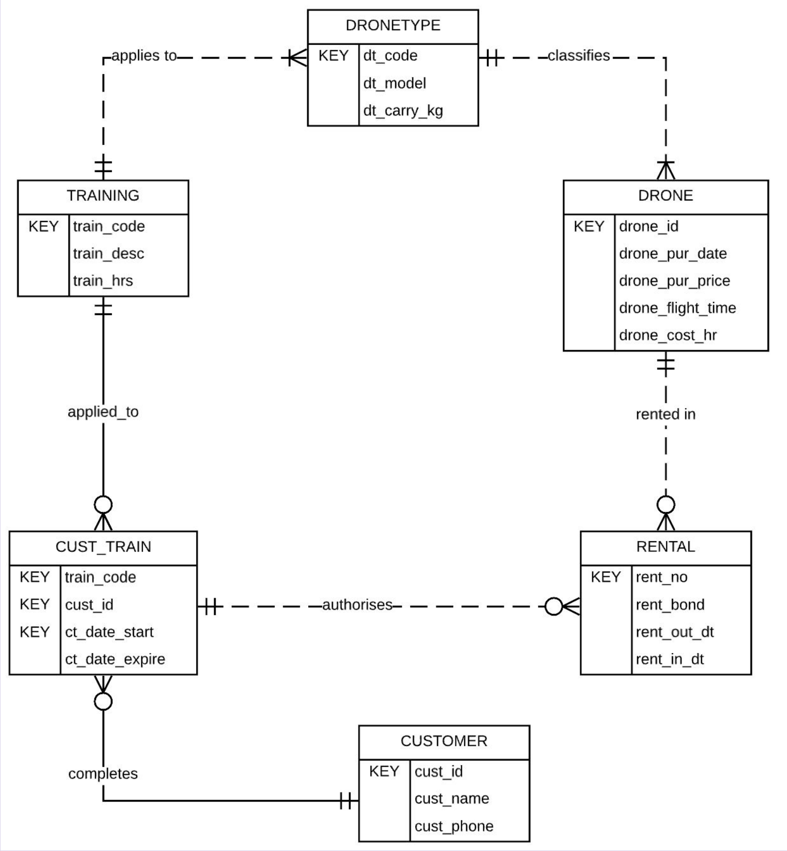


EMPLOYEE		
P	*	emp_id
	*	emp_name

EMP_DEPENDENT		
PF	*	emp_id
P	*	edep_id
	*	edep_firstname
		edep_middleinit
	*	edep_lastname
	*	edep_dob
	*	edep_gender

Conceptual

Logical



DRONETYPE	
KEY	dt_code
	dt_model
	dt_carry_kg

TRAINING	
KEY	train_code
	train_desc
	train_hrs

DRONE	
KEY	drone_id
	drone_pur_date
	drone_pur_price
	drone_flight_time
	drone_cost_hr

CUST_TRAIN	
KEY	train_code
KEY	cust_id
KEY	ct_date_start
	ct_date_expire

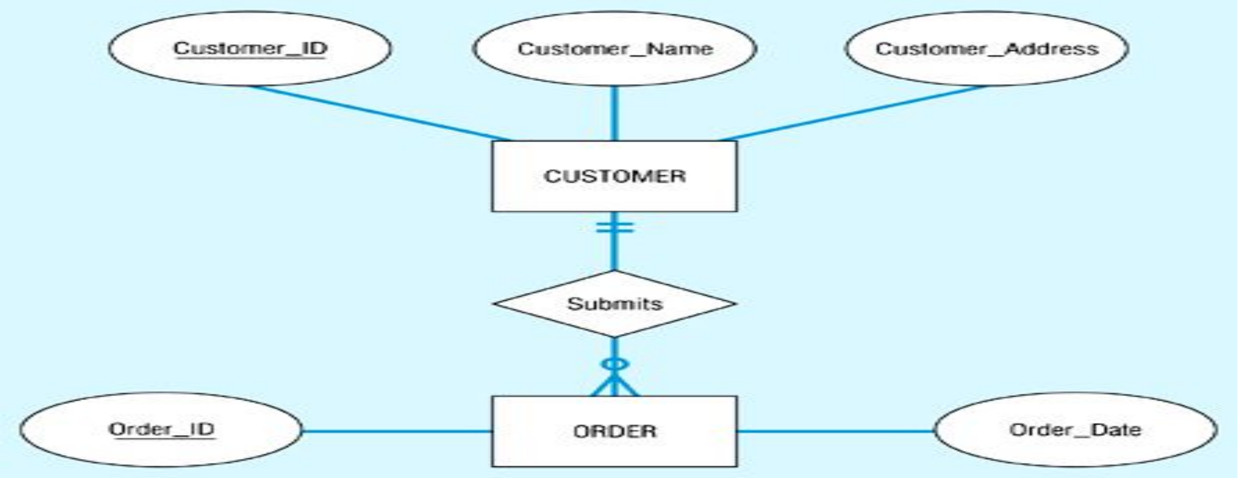
RENTAL	
KEY	rent_no
	rent_bond
	rent_out_dt
	rent_in_dt

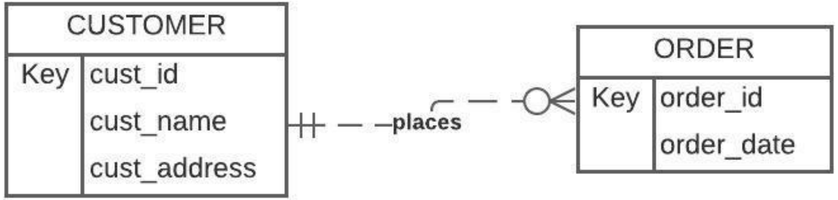
CUSTOMER	
KEY	cust_id
	cust_name
	cust_phone

As a group map:

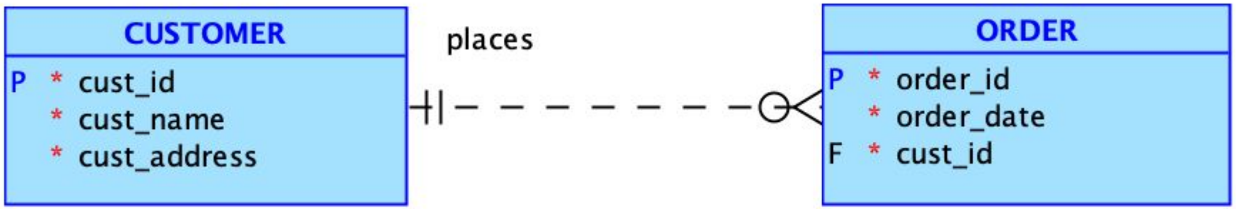
- **entities**
- **keys**
- **attributes (allow nulls?)**

Mapping a 1:M Binary Relationship



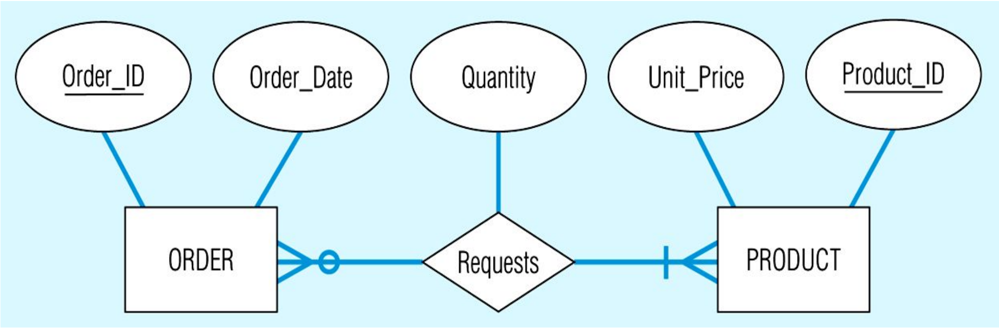


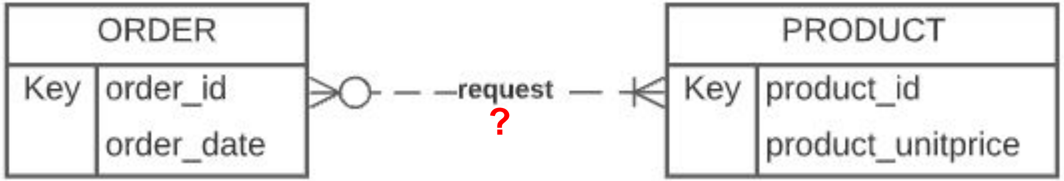
Map Binary Relationships (1:M)



For each 1:M binary relationship, first create a relation for each of the two entity types participating in the relationship. Then include the PK attribute (or attributes) of the entity on the one-side of the relationship as the FK on the many-side of the relationship.

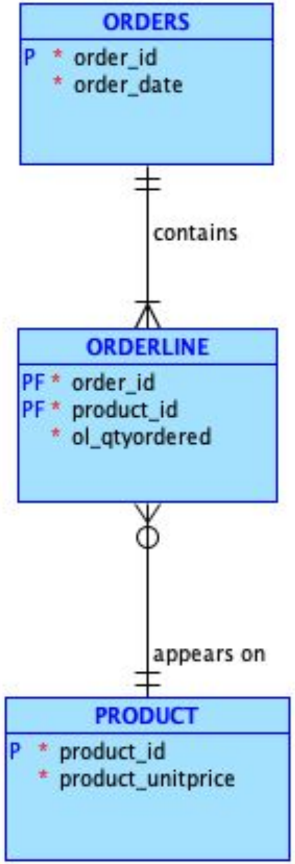
Mapping a M:N Binary Relationship





ORDER	
Key	order_id
	order_date

PRODUCT	
Key	product_id
	product_unitprice



ORDERS

P * order_id
* order_date

contains

ORDERLINE

PF * order_id
PF * product_id
* ol_qtyordered

appears on

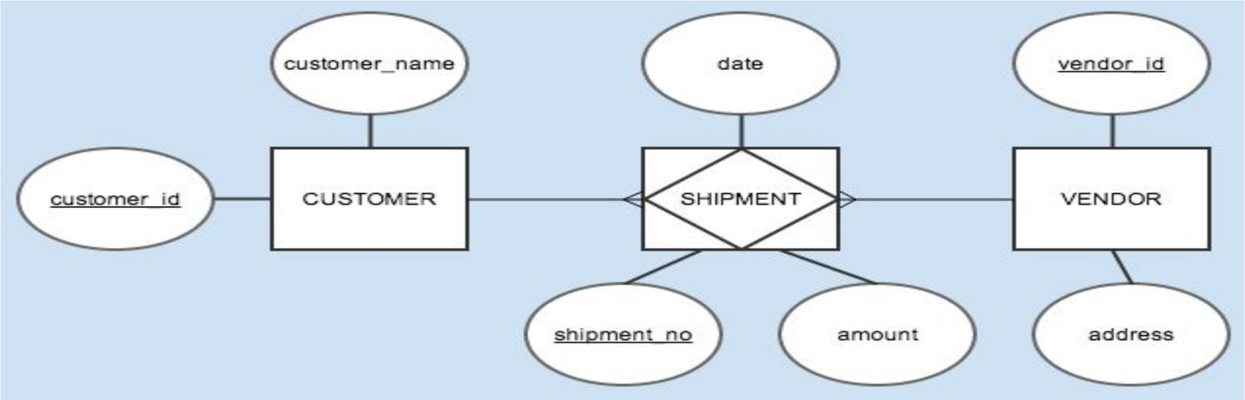
PRODUCT

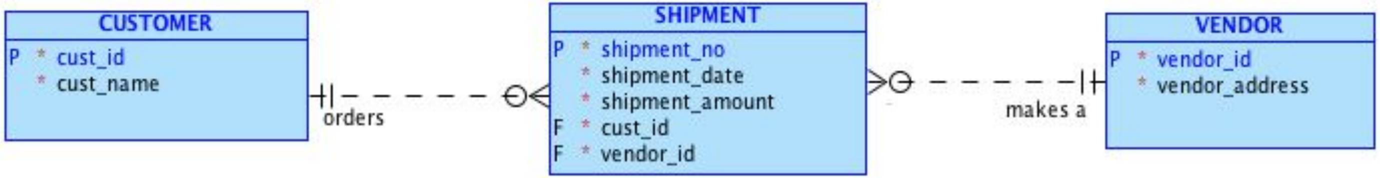
P * product_id
* product_unitprice

Map Binary Relationship (M:N)

- For a M:N binary relationship
  - First create a relation for each of the two entity types participating in the relationship.
  - Then create a new relation and include as foreign key attributes, the PK attribute (or attributes) for each of the two participating entity types. These attributes become the PK of the new relation.
  - If there are any nonkey attributes associated with the M:N relationship, they are also included in the new relation.

Mapping an associative entity with an Identifier



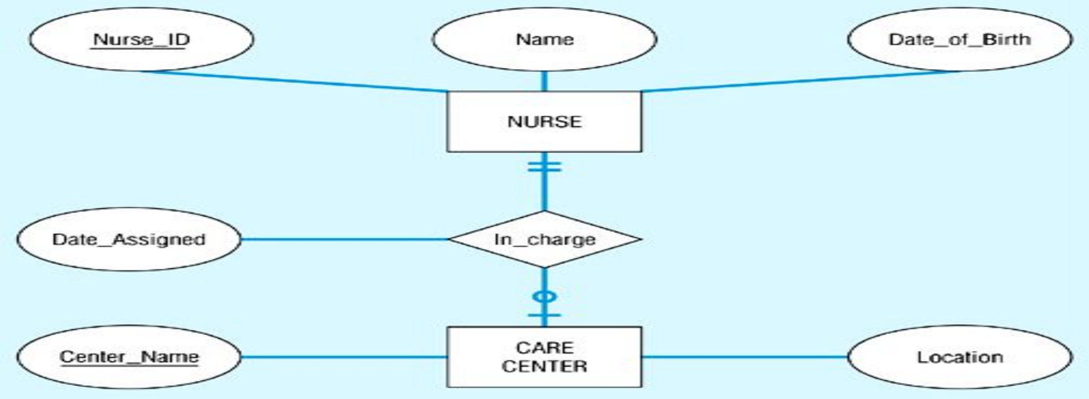


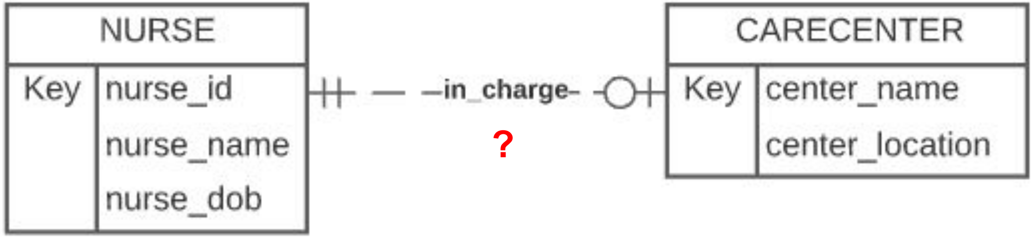
CUSTOMER
P * cust_id
* cust_name

SHIPMENT
P * shipment_no
* shipment_date
* shipment_amount
F * cust_id
F * vendor_id

VENDOR
P * vendor_id
* vendor_address

Mapping a 1:1 Binary Relationship





NURSE	
Key	nurse_id
	nurse_name
	nurse_dob

CARECENTER	
Key	center_name
	center_location

?

Map Binary Relationship (1:1)

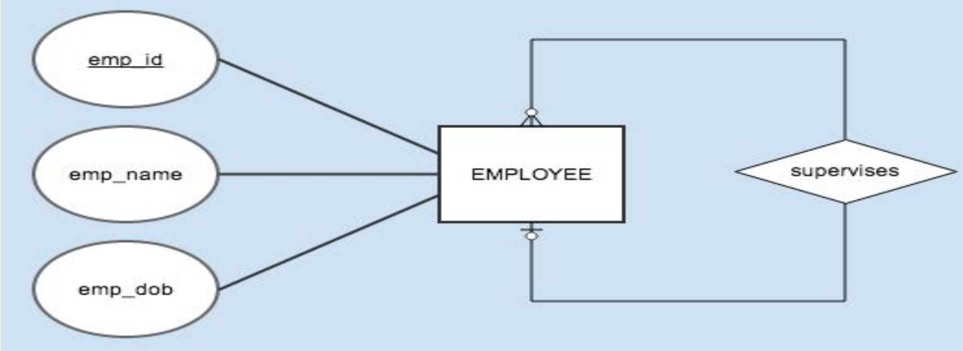
- Create two relations, one for each of the participating entity types.
  - The primary key (PK) on the mandatory side of the relationship becomes the foreign key (FK) on the optional side of the relationship.
    - -+|- Mandatory side
    - -O+ Optional side

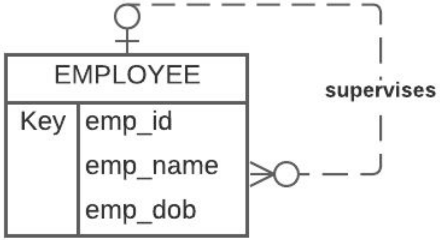
    Note here mandatory/optional as used above is not speaking about participation, here we are referring to minimum cardinality
  - where both are optional place the FK on the side which causes the fewest nulls
  - Special case: 1:1 total relationship (mandatory participation from both sides)
    - Should consolidating the two entity types into one relation

Map unary relationships

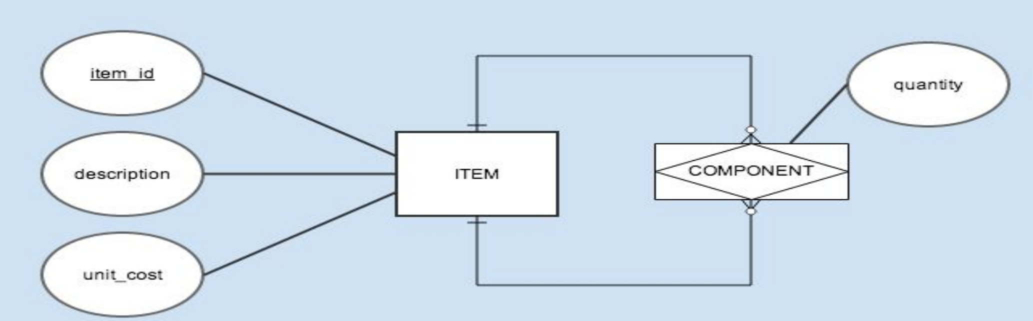
- Unary Relationship is a relationship between the instances of a single entity type.
- Unary 1:M Relationship – A relation is created for the entity type. Add a FK within the same relation that references the PK of the relation. A recursive foreign key is a FK in a relation that references the PK values of the same relation.
- Unary M:N Relationship – Two relations are created, one for the entity type in the relationship and the other as the associative relation to represent the M:N relationship itself. The PK of the associative relation consists of two attributes (with different names) taking their values from the PK of the other relation.

Mapping a 1:M Unary Relationship

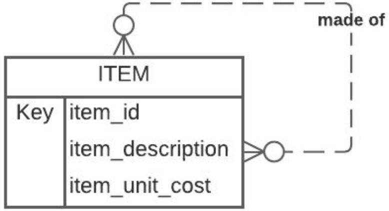


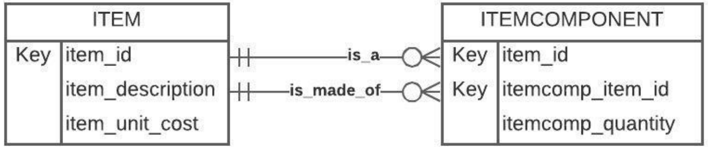


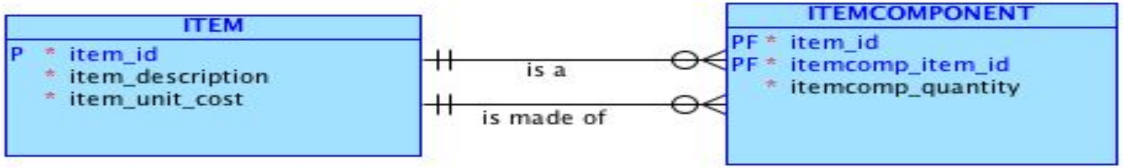
Mapping a M:N Unary Relationship



quantity?

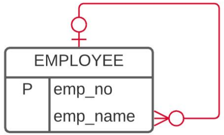






IMPORTANT NOTE

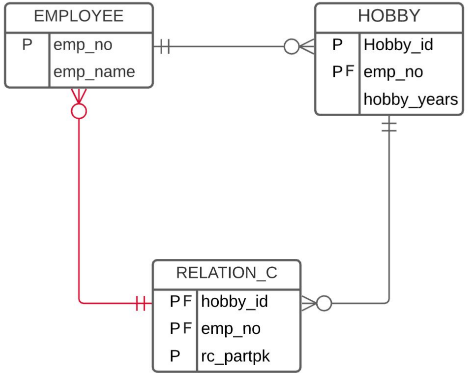
- Apply the principles we have discussed, think carefully about the consequences of relationship placement
  - Recursive identifying relationships CANNOT exist
  - 1:1 Total identifying relationships CANNOT exist
  - Take care with relationship "loops"
- What happens here:



EMPLOYEE

P emp_no

emp_name



EMPLOYEE

P emp_no

emp_name

HOBBY

P Hobby_id

PF emp_no

hobby_years

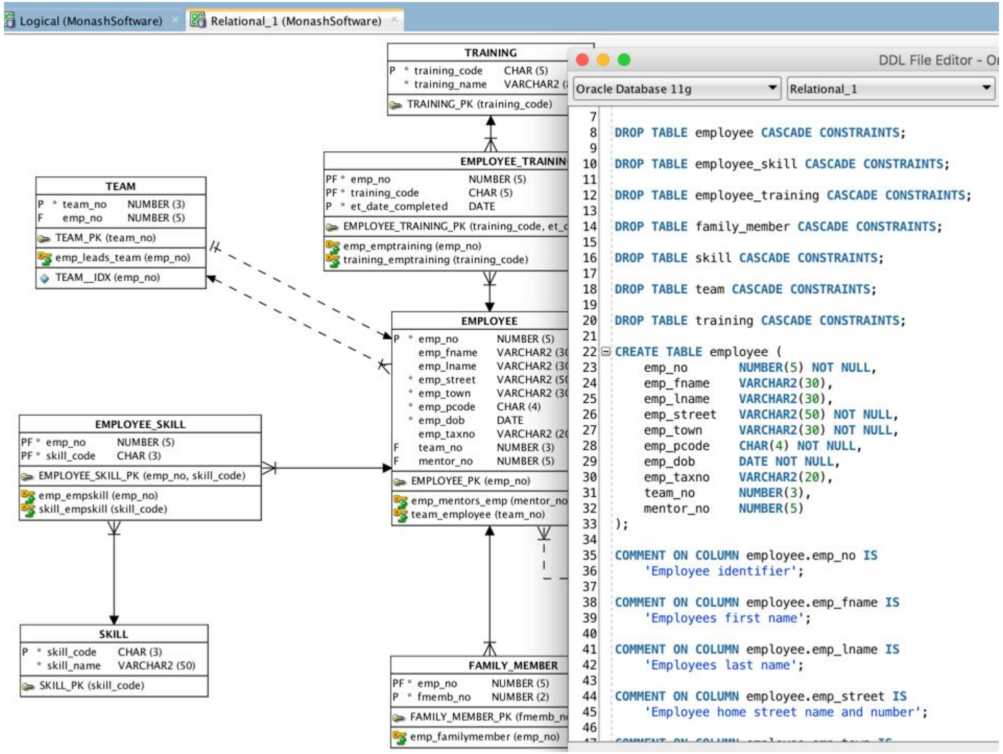
RELATION_C

PF hobby_id

PF emp_no

P rc_partpk

Oracle Data Modeler



TRAINING
P * training_code CHAR (5)
* training_name VARCHAR2 (
TRAINING_PK (training_code)

EMPLOYEE_TRAININ
PF * emp_no NUMBER (5)
PF * training_code CHAR (5)
P * et_date_completed DATE
EMPLOYEE_TRAINING_PK (training_code, et_
emp_emptraining (emp_no)
training_emptraining (training_code)

TEAM
P * team_no NUMBER (3)
F emp_no NUMBER (5)
TEAM_PK (team_no)
emp_leads_team (emp_no)
TEAM__IDX (emp_no)

EMPLOYEE
P * emp_no NUMBER (5)
emp_fname VARCHAR2 (3
emp_lname VARCHAR2 (3
* emp_street VARCHAR2 (5
* emp_town VARCHAR2 (3
* emp_pcode CHAR (4)
* emp_dob DATE
emp_taxno VARCHAR2 (2
F team_no NUMBER (3)
F mentor_no NUMBER (5)
EMPLOYEE_PK (emp_no)
emp_mentors_emp (mentor_no
team_employee (team_no)

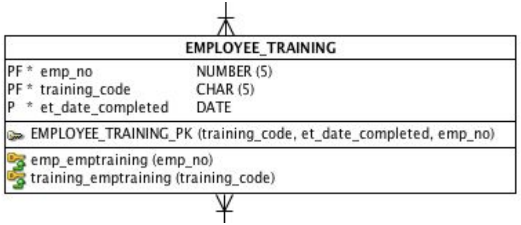
EMPLOYEE_SKILL
PF * emp_no NUMBER (5)
PF * skill_code CHAR (3)
EMPLOYEE_SKILL_PK (emp_no, skill_code)
emp_empskill (emp_no)
skill_empskill (skill_code)

SKILL
P * skill_code CHAR (3)
* skill_name VARCHAR2 (50)
SKILL_PK (skill_code)

FAMILY_MEMBER
PF * emp_no NUMBER (5)
P * fmemb_no NUMBER (2)
FAMILY_MEMBER_PK (fmemb_n
emp_familymember (emp_no)

```
DROP TABLE employee CASCADE CONSTRAINTS;

DROP TABLE employee_skill CASCADE CONSTRAINTS;

DROP TABLE employee_training CASCADE CONSTRAINTS;

DROP TABLE family_member CASCADE CONSTRAINTS;

DROP TABLE skill CASCADE CONSTRAINTS;

DROP TABLE team CASCADE CONSTRAINTS;

DROP TABLE training CASCADE CONSTRAINTS;

CREATE TABLE employee (
    emp_no        NUMBER(5) NOT NULL,
    emp_fname     VARCHAR2(30),
    emp_lname     VARCHAR2(30),
    emp_street    VARCHAR2(50) NOT NULL,
    emp_town      VARCHAR2(30) NOT NULL,
    emp_pcode     CHAR(4) NOT NULL,
    emp_dob       DATE NOT NULL,
    emp_taxno     VARCHAR2(20),
    team_no       NUMBER(3),
    mentor_no     NUMBER(5)
);

COMMENT ON COLUMN employee.emp_no IS
    'Employee identifier';

COMMENT ON COLUMN employee.emp_fname IS
    'Employees first name';

COMMENT ON COLUMN employee.emp_lname IS
    'Employees last name';

COMMENT ON COLUMN employee.emp_street IS
    'Employee home street name and number';

COMMENT ON COLUMN employee.emp_town IS
```

Adding surrogate keys



```
EMPLOYEE_TRAINING
PF * emp_no              NUMBER (5)
PF * training_code       CHAR (5)
P  * et_date_completed   DATE
EMPLOYEE_TRAINING_PK (training_code, et_date_completed, emp_no)
emp_emptraining (emp_no)
training_emptraining (training_code)
```

Surrogate PK's may be added **ONLY** on the logical model provided they are justified

MANUALLY add new PK attribute (here **et_no**), **DO NOT USE** Modelers "Create Surrogate Key" option

Potential problem:

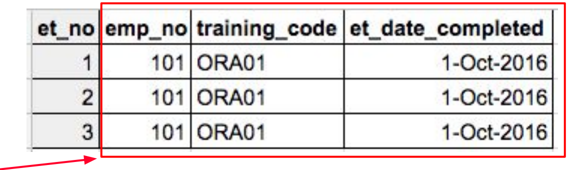
Need to ensure that the identified key from the conceptual model - the natural key:

(emp_no, training_code, et_date_completed)

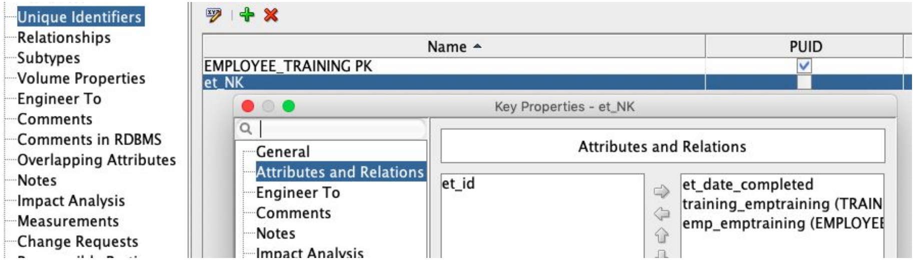
will still remain unique

Solution:

Define a unique index on the attributes of natural key



et_no	emp_no	training_code	et_date_completed
1	101	ORA01	1-Oct-2016
2	101	ORA01	1-Oct-2016
3	101	ORA01	1-Oct-2016

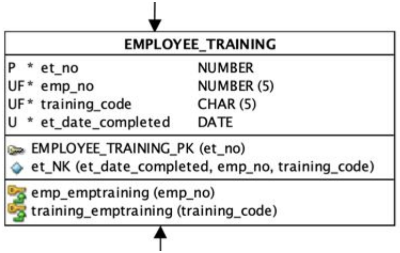


Name ▲	PUID
EMPLOYEE_TRAINING PK	☑
et_NK	□

Key Properties - et_NK

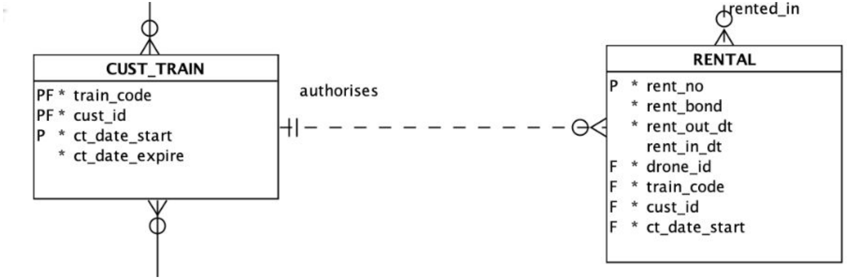
et_id

et_date_completed
training_emptraining (TRAIN
emp_emptraining (EMPLOYEE



```
EMPLOYEE_TRAINING
P  * et_no               NUMBER
UF * emp_no              NUMBER (5)
UF * training_code       CHAR (5)
U  * et_date_completed   DATE
EMPLOYEE_TRAINING_PK (et_no)
et_NK (et_date_completed, emp_no, training_code)
emp_emptraining (emp_no)
training_emptraining (training_code)
```

Adding surrogate keys continued



CUST_TRAIN

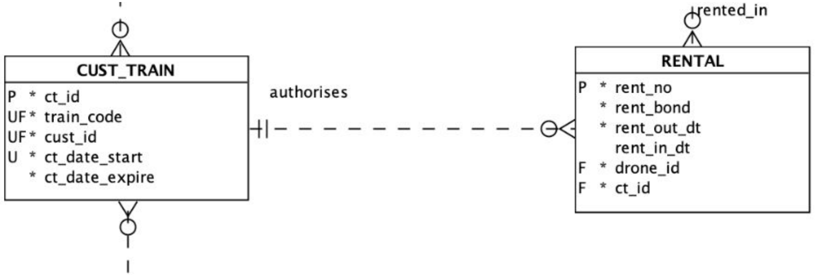
PF * train_code
PF * cust_id
P * ct_date_start
* ct_date_expire

authorises

RENTAL

P * rent_no
* rent_bond
* rent_out_dt
rent_in_dt
F * drone_id
F * train_code
F * cust_id
F * ct_date_start

rented_in



CUST_TRAIN

P * ct_id
UF * train_code
UF * cust_id
U * ct_date_start
* ct_date_expire

authorises

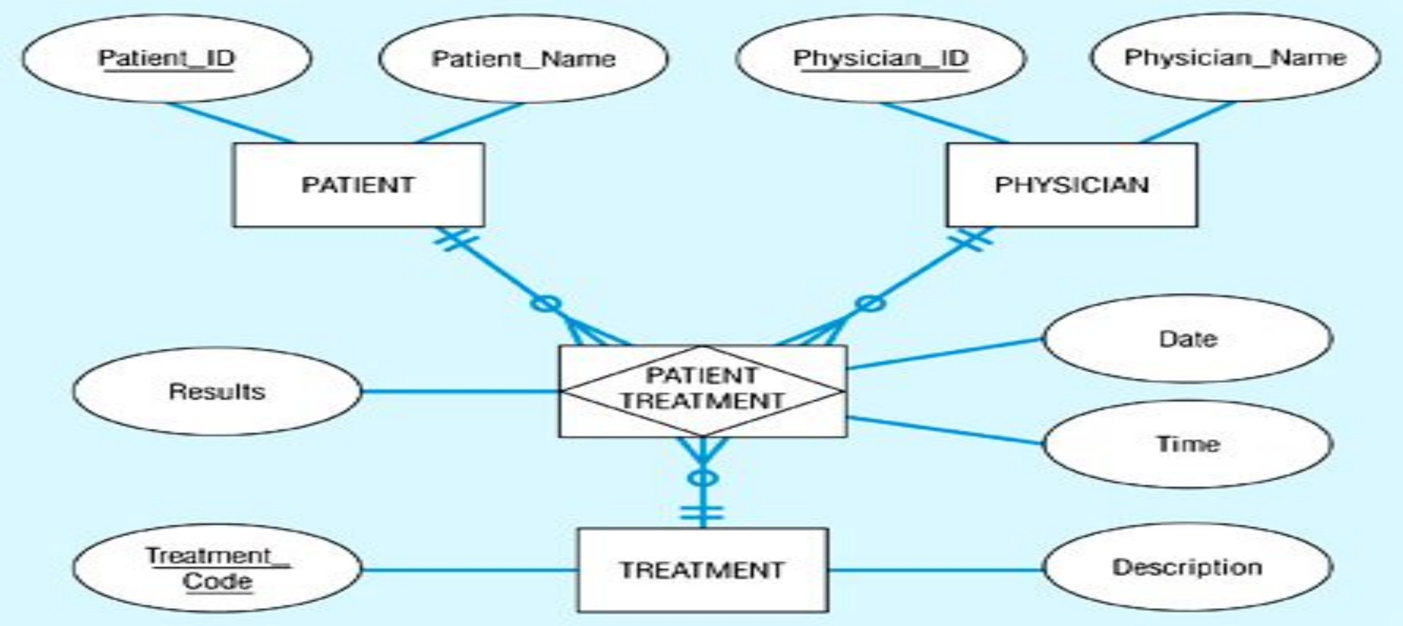
RENTAL

P * rent_no
* rent_bond
* rent_out_dt
rent_in_dt
F * drone_id
F * ct_id

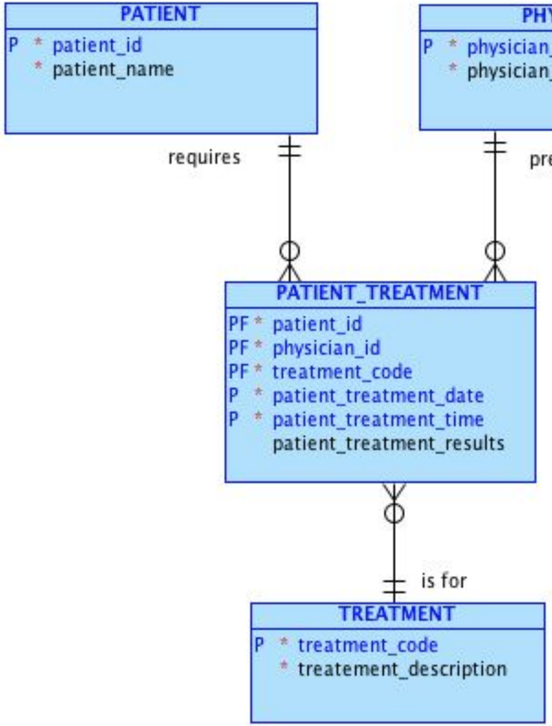
rented_in

- As a guiding principle, surrogate keys must be added where there are more than two attributes in a primary key
  - some organisations add surrogates to all relations since they are
    - a single numeric attribute, and
    - have no inherent meaning
- **When adding a surrogate key you MUST protect the natural key via a unique index**
- In a model adding a surrogate:
  - simplifies a PK which can address design issues/problems
    - CUST_TRAIN: PK train_code, cust_id, ct_date_start => ct_id
  - simplifies FK's
    - RENTAL: FK train_code, cust_id, ct_date_start => ct_id

Mapping a Ternary Relationship



Mapping a Ternary Relationship



Overall Design Process - checklist

Assignment 1

Step 1: **complete a conceptual model based on scenario described in Task 1**

Step 2: normalise the supplied form in Task 2 to 3NF, UNF->1NF->2NF->3NF

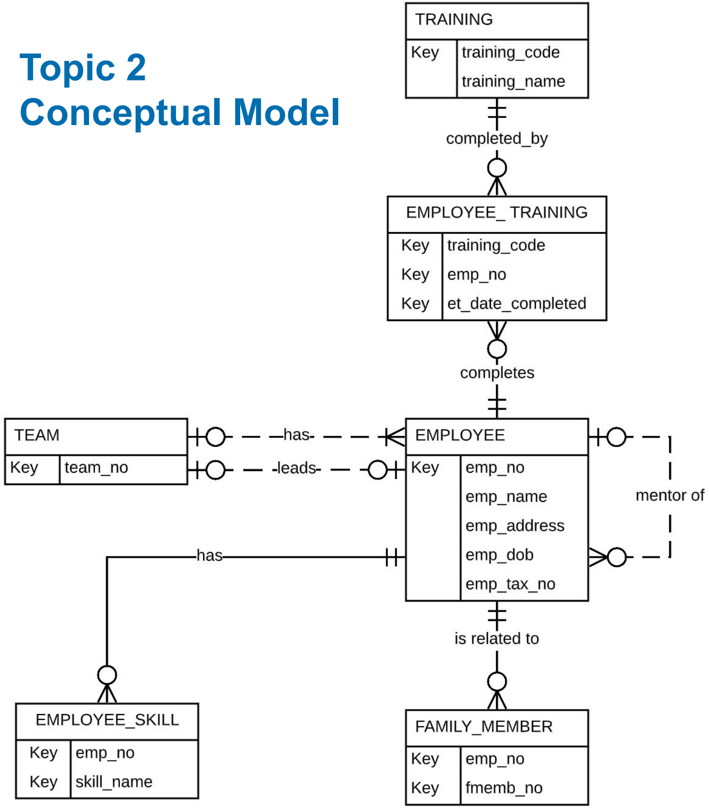
- Carry out attribute synthesis, if required, on resultant set of 3NF relations to obtain final set of 3NF relations

Step 3: map your conceptual model to a relational logical model

- Integrate your final set of 3NF relations from Step 2 above, ensure attribute names in your normalisation are consistent with the names used in your logical model
- add in any new attributes detailed in the **Task 3** scenario
- check model for:
  - no insert/update/delete anomalies
  - surrogate key requirement (if added ensure unique constraint/index on natural key created)
  - the need for check clauses/lookup tables
- generate physical (relational) model and from this generate the schema file, add appropriate details to the schema file as detailed in the brief (see applied 6)
- run the schema file in Oracle and ensure no errors occur (if there are any go back and fix logical model and repeat generation/run until no errors occur)

Monash Software Conceptual Model

Topic 2
Conceptual Model



Topic 4 Normalisation

3NF

EMPLOYEE (emp_no, emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode)

EMP_PHONE (emp_no, phone_no, phone_type)

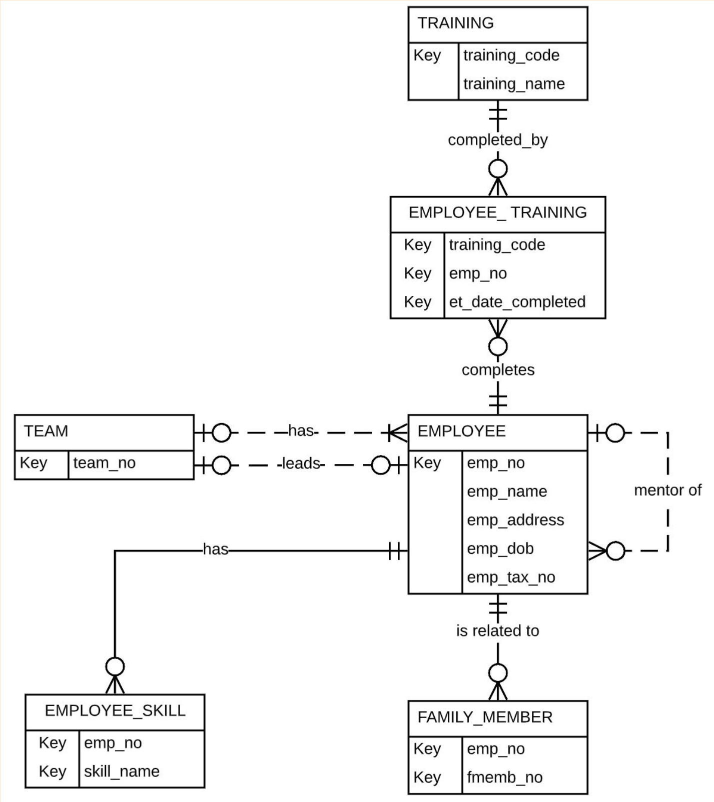
EMP_QUALIFICATION (emp_no, degree_name, degree_institution, degree_year)

FAMILY_MEMBER (emp_no, fmemb_no, fmemb_name, fmemb_dob)

EMPLOYEE_SKILL (emp_no, skill_name)

Post Workshop Task - answer available Sunday 5 PM

Convert this model to a logical model and then integrate the normalisation from the Topic 4 Workshop


TRAINING
Key training_code
training_name
completed_by
EMPLOYEE_ TRAINING
Key training_code
Key emp_no
Key et_date_completed
completes
TEAM
Key team_no
has
leads
EMPLOYEE
Key emp_no
emp_name
emp_address
emp_dob
emp_tax_no
mentor of
has
is related to
EMPLOYEE_SKILL
Key emp_no
Key skill_name
FAMILY_MEMBER
Key emp_no
Key fmemb_no