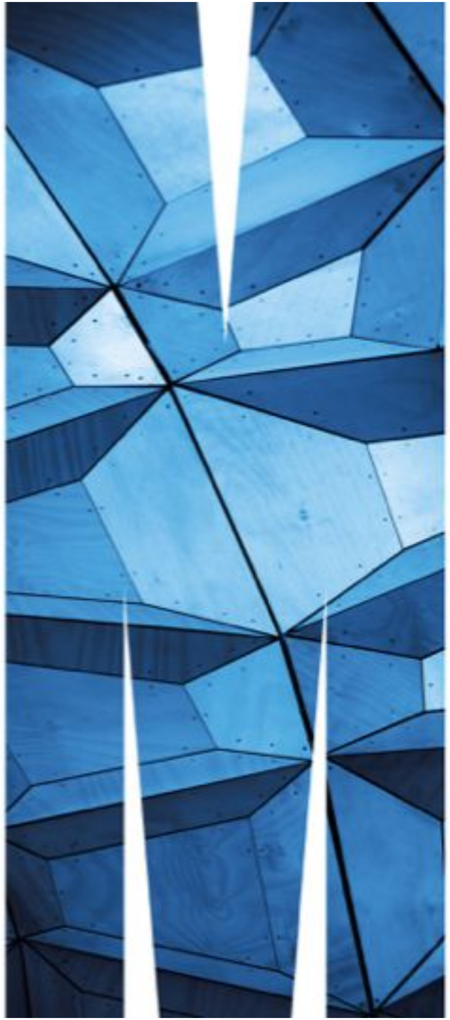
MONASH
University

MONASH
INFORMATION
TECHNOLOGY

Database Design II: Logical Modelling



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 (week 2)
  - Database Model independent
- Step 2 Logical Model (this week)
  - 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	Primary Key	Primary Key
Relationship	---	---
---	Foreign Key	Foreign Key

Q1. Which of the following are invalid relations (note your reasons as you make your decisions):

A. EMPLOYEE (empname, empdept, empno, empsalary)

B. CUSTOMER (custno, custname, custphone, custphone)

C. ORDER (orderno, orderdate, custno)

D. PRODUCT (prodno, proddesc, produprice)

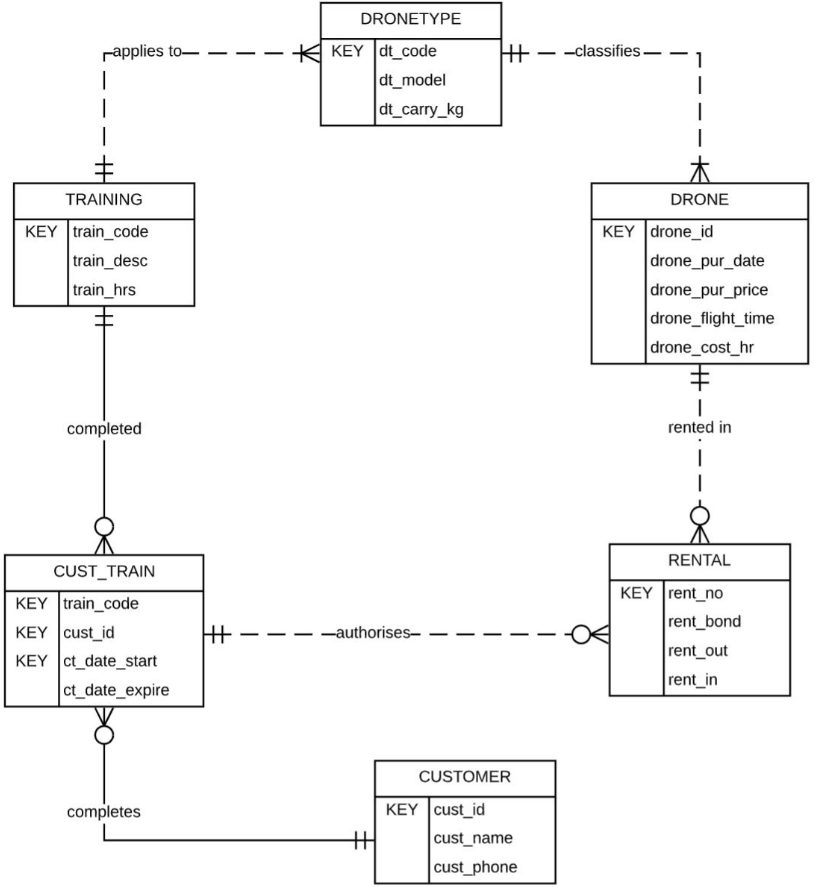
E. TRIP (trip_id, driver_id, driver_name, (stop_id, stop_time))

F. LICENCE_HELD (driver_id, licence_type)

Recap Week 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

HiFlying Drone 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
	rent_in

CUSTOMER

KEY	cust_id
	cust_name
	cust_phone

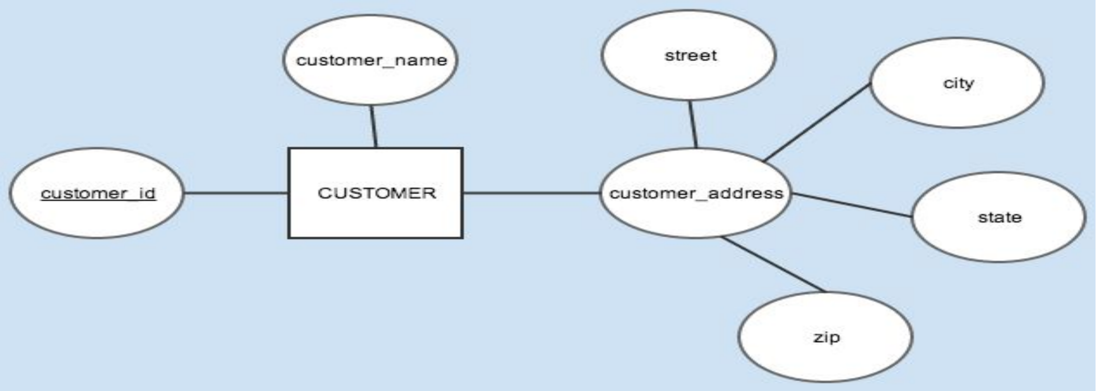
Transforming ER diagrams into relations (mapping conceptual level to logical level)

- 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).*

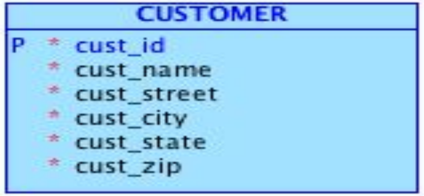
Map Regular Entities

- 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, e.g. phone numbers

Mapping a Composite Attribute



CUSTOMER	
Key	cust_id
	cust_name
	cust_street
	cust_city
	cust_state
	cust_zip



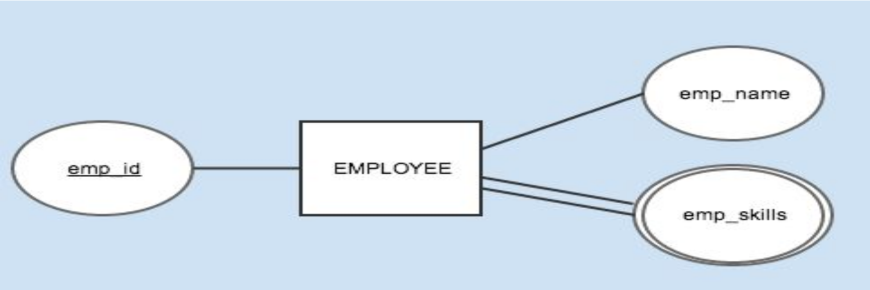
CUSTOMER

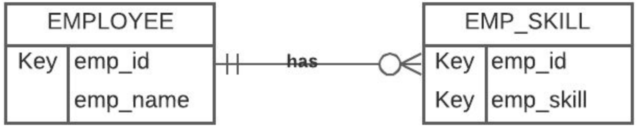
P * cust_id
* cust_name
* cust_street
* cust_city
* cust_state
* cust_zip

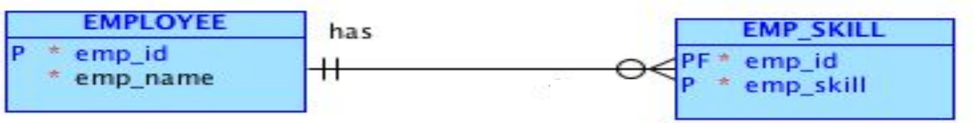
Map Regular Entities

- 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



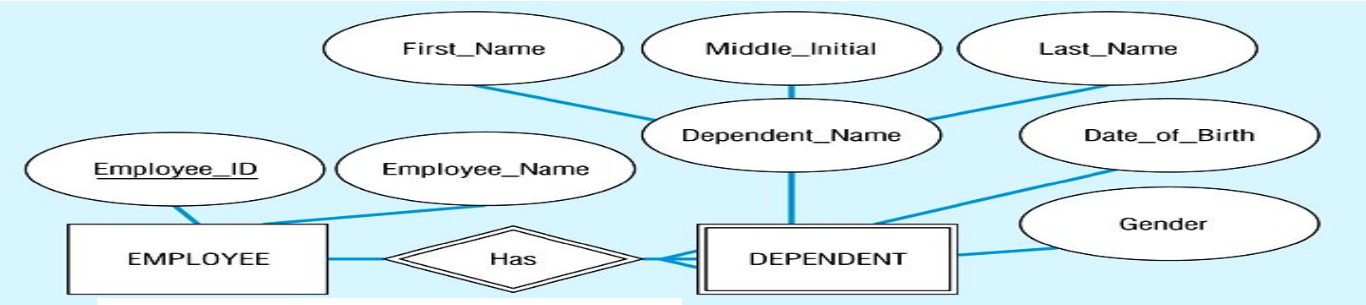


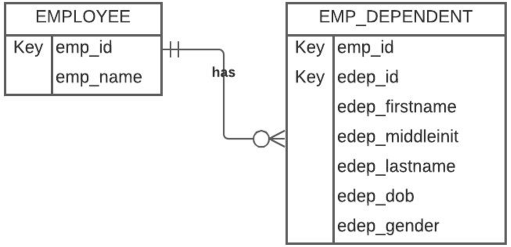


Is there a better solution than the one shown above?
What are the issues here?

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.

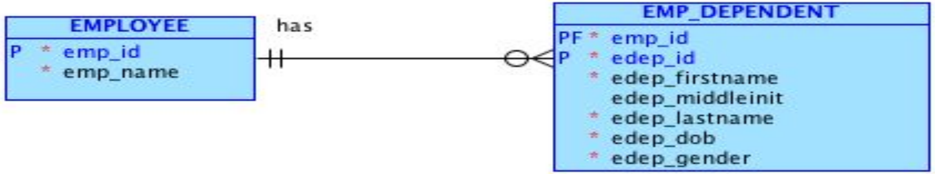




EMPLOYEE	
Key	emp_id
	emp_name

has

EMP_DEPENDENT	
Key	emp_id
Key	edep_id
	edep_firstname
	edep_middleinit
	edep_lastname
	edep_dob
	edep_gender

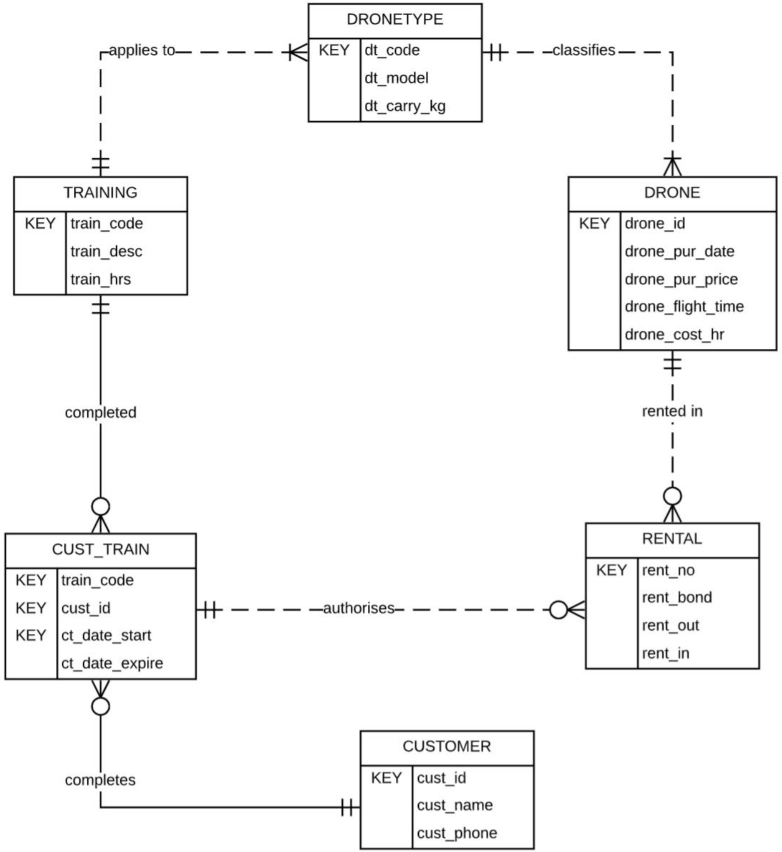


EMPLOYEE		
P	*	emp_id
	*	emp_name

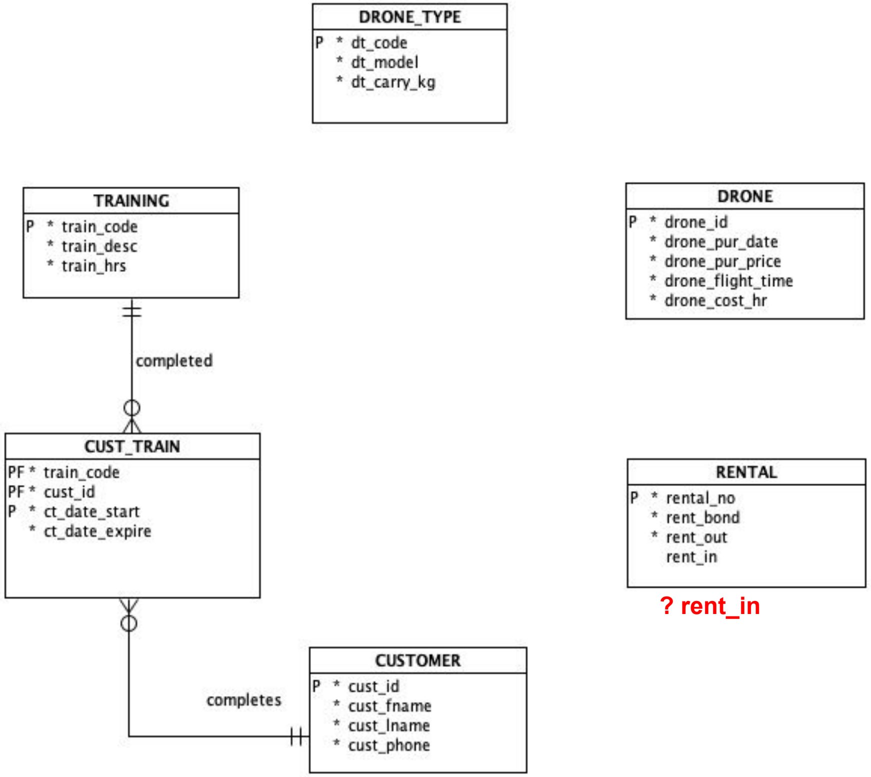
has

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

Conceptual



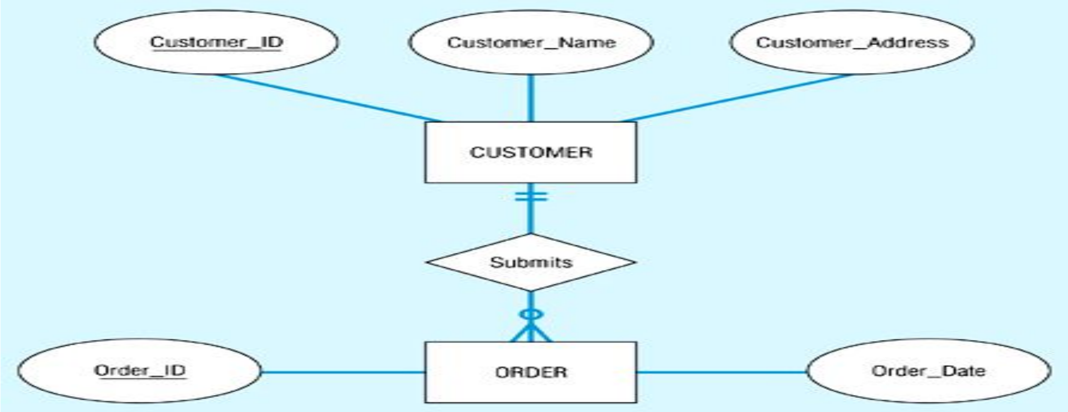
Logical



? rent_in

? phone

Mapping a 1:M Binary Relationship



Customer_ID

Customer_Name

Customer_Address

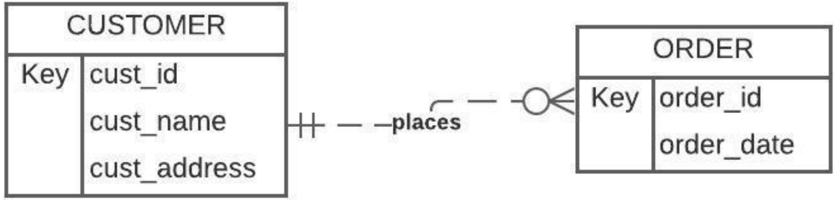
CUSTOMER

Submits

Order_ID

ORDER

Order_Date

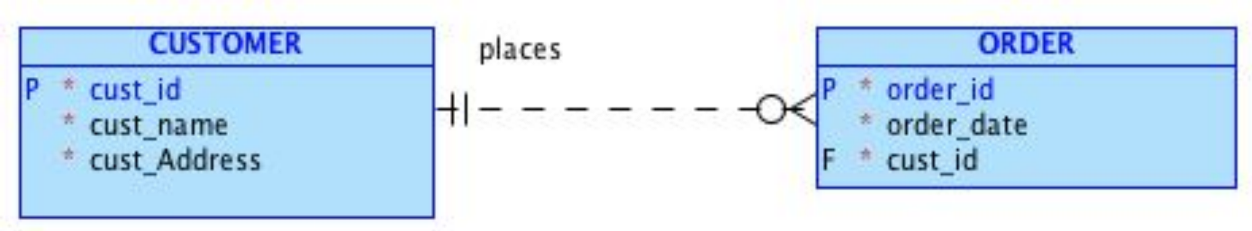


CUSTOMER	
Key	cust_id
	cust_name
	cust_address

places

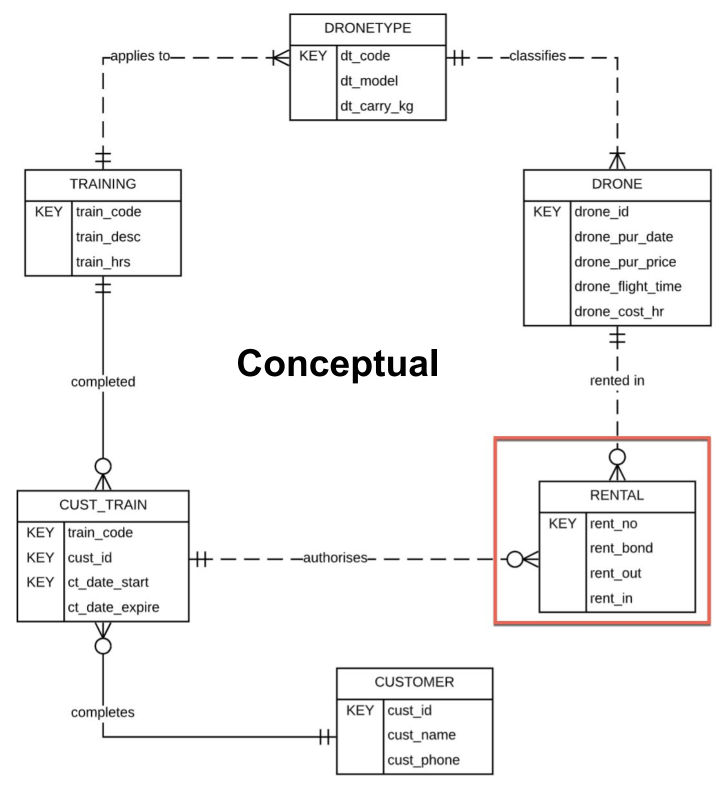
ORDER	
Key	order_id
	order_date

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.

Q2 The full set of attributes that are required in the RENTAL relation will be:



A. rent_no, rent_bond, rent_out, rent_in

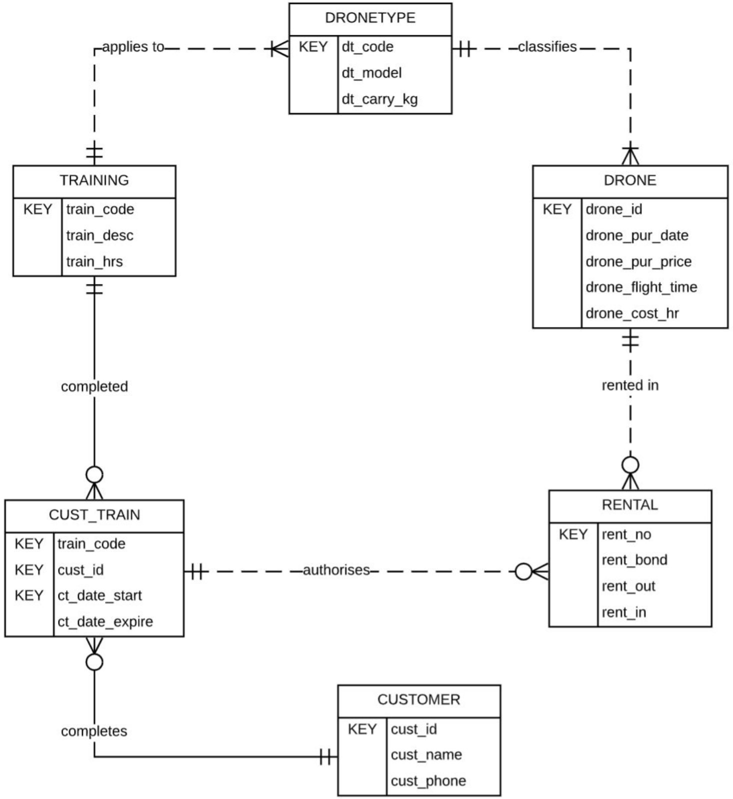
B. rent_no, rent_bond, rent_out, rent_in, drone_id

C. rent_no, rent_bond, rent_out, rent_in, train_code, cust_id, ct_date_start, drone_id

D. rent_no, rent_bond, rent_out, rent_in, train_code, drone_id

E. rent_no, rent_bond, rent_out, rent_in, train_code, cust_id, ct_date_start, drone_id, dr_code

Conceptual



applies to

classifies

completed

rented in

authorises

completes

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

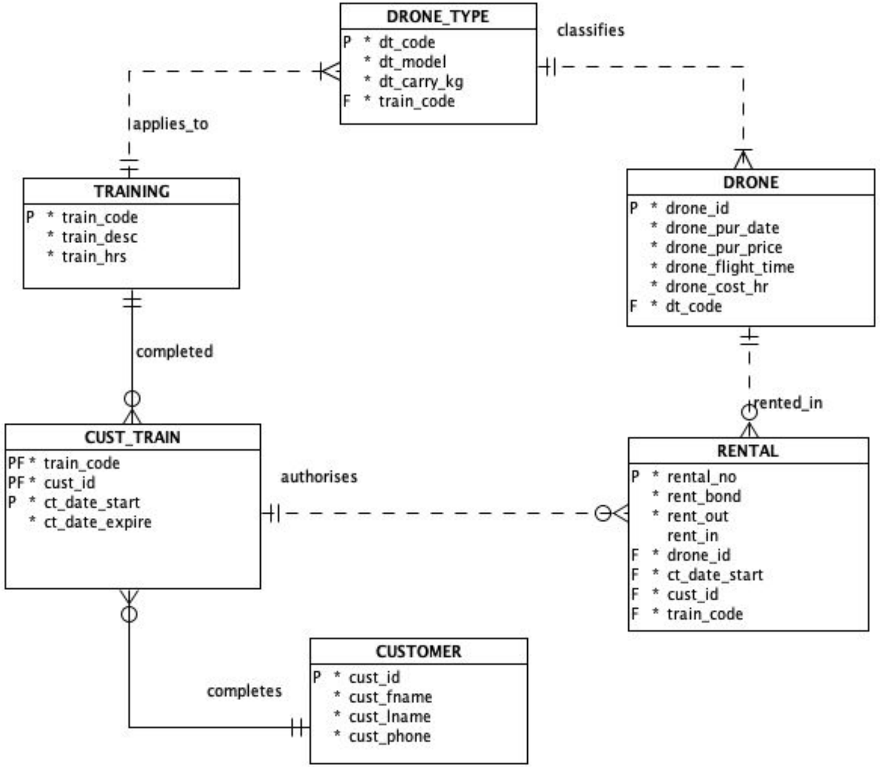
RENTAL

KEY	rent_no
	rent_bond
	rent_out
	rent_in

CUSTOMER

KEY	cust_id
	cust_name
	cust_phone

Logical



classifies

applies_to

completed

rented_in

authorises

completes

DRONE_TYPE

P	*	dt_code
	*	dt_model
	*	dt_carry_kg
F	*	train_code

TRAINING

P	*	train_code
	*	train_desc
	*	train_hrs

DRONE

P	*	drone_id
	*	drone_pur_date
	*	drone_pur_price
	*	drone_flight_time
	*	drone_cost_hr
F	*	dt_code

CUST_TRAIN

PF	*	train_code
PF	*	cust_id
P	*	ct_date_start
	*	ct_date_expire

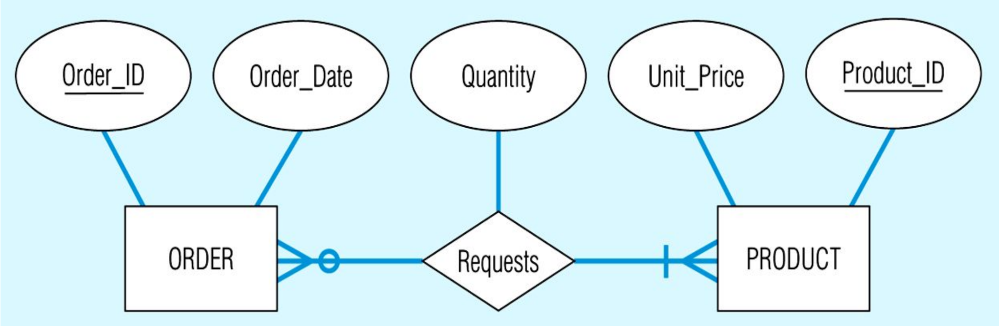
RENTAL

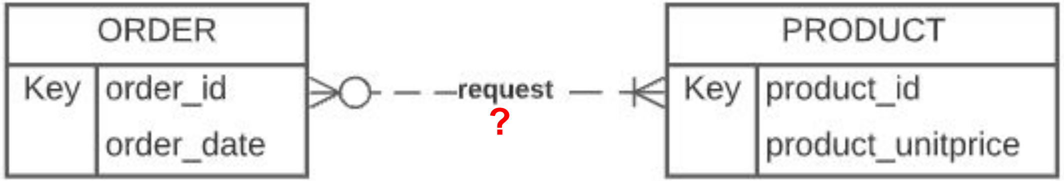
P	*	rental_no
	*	rent_bond
	*	rent_out
		rent_in
F	*	drone_id
F	*	ct_date_start
F	*	cust_id
F	*	train_code

CUSTOMER

P	*	cust_id
	*	cust_fname
	*	cust_lname
	*	cust_phone

Mapping a M:N Binary Relationship



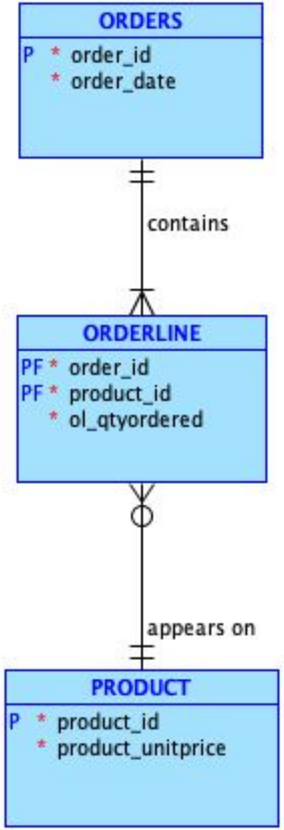


ORDER	
Key	order_id
	order_date

request

?

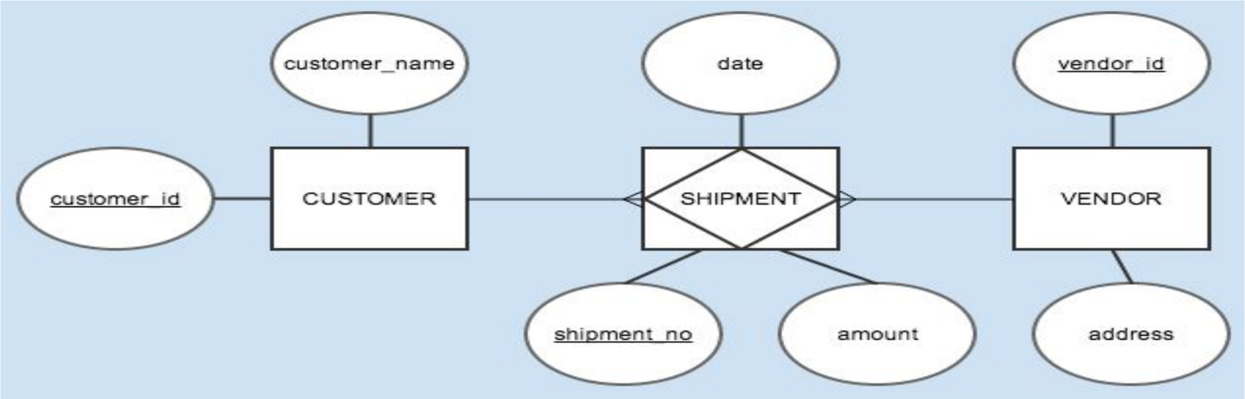
PRODUCT	
Key	product_id
	product_unitprice

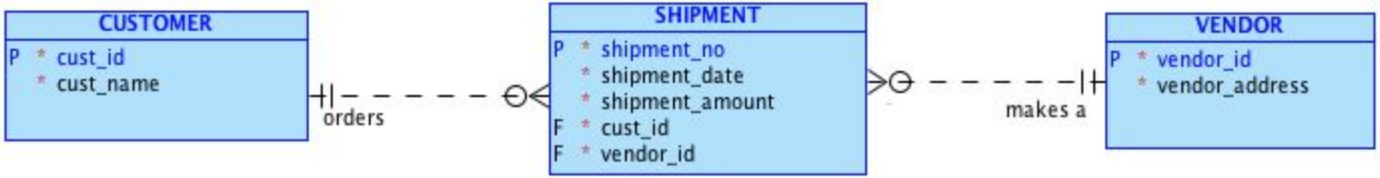

ORDERS
P * order_id
* order_date
contains
ORDERLINE
PF * order_id
PF * product_id
* ol_qtyordered
appears on
PRODUCT
P * product_id
* product_unitprice

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

orders

SHIPMENT

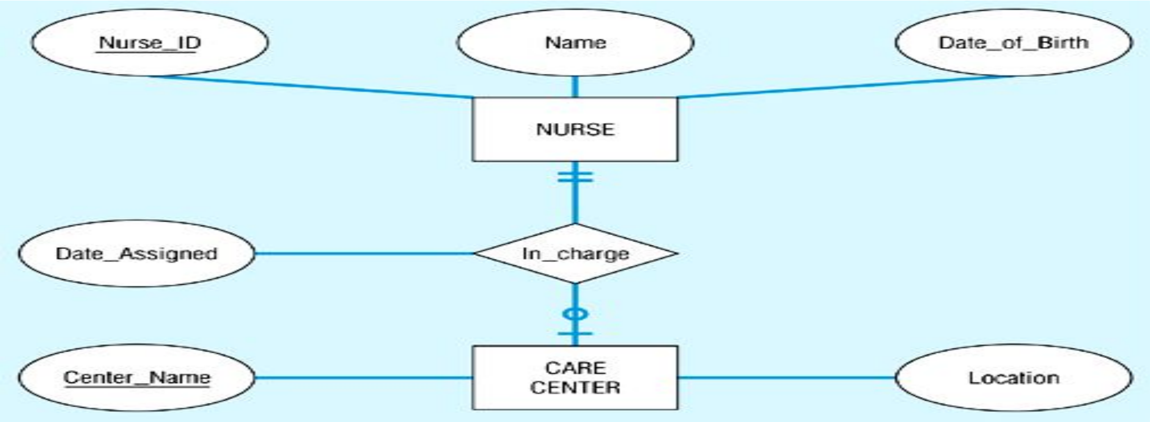
P * shipment_no
* shipment_date
* shipment_amount
F * cust_id
F * vendor_id

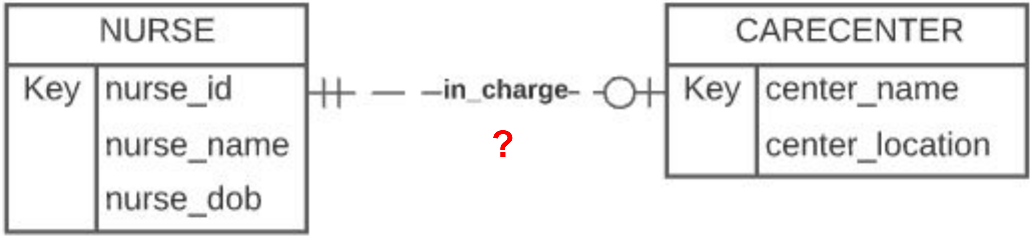
makes a

VENDOR

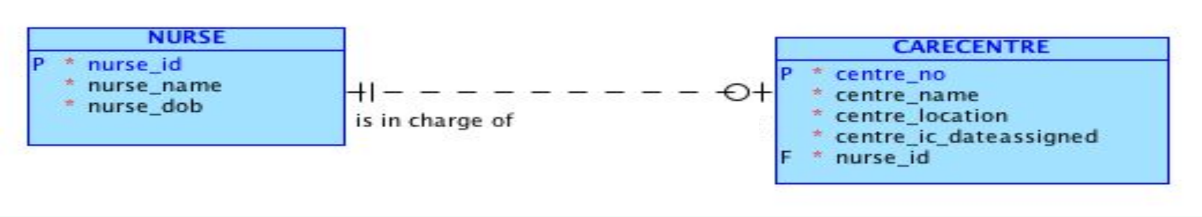
P * vendor_id
* vendor_address

Mapping a 1:1 Binary Relationship





Q3. NURSE participation in this relationship is:



A. Optional
B. Mandatory
C. It depends on the implementation
D. 1:1
E. 0

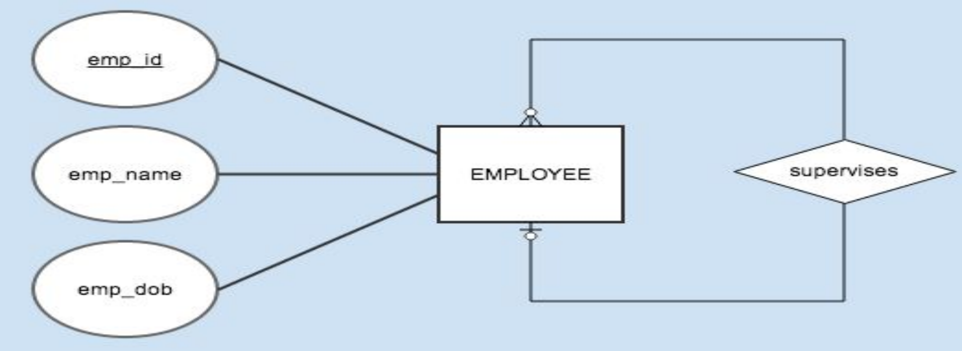
Map Binary Relationship (1:1)

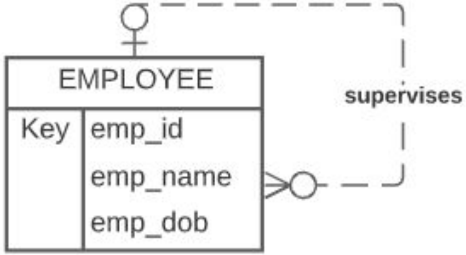
- Create two relations, one for each of the participating entity types.
  - The primary key (PK) on the mandatory side of the relationship becomes the foreign key (FK) on the optional side of the relationship.
  - where both are optional place the FK on the side which causes the fewest nulls
  - Special case: 1:1 total relationship (mandatory participation from both sides)
    - Should consolidating the two entity types into one relation

Map unary relationships

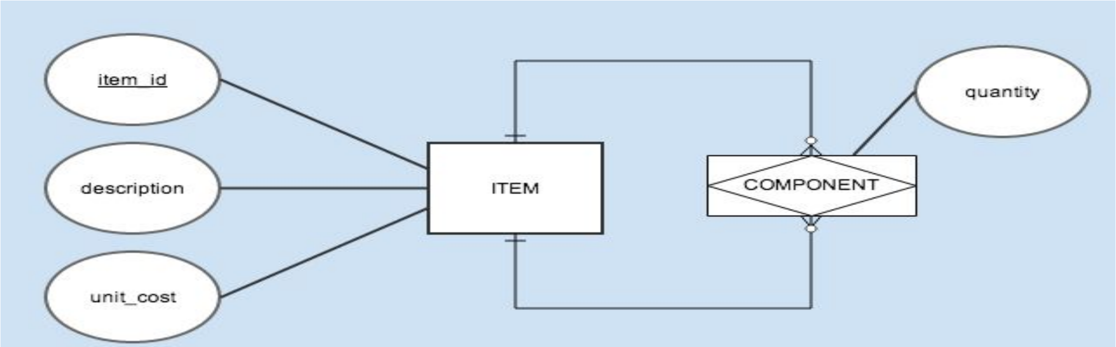
- Unary Relationship is a relationship between the instances of a single entity type.
- Unary 1:M Relationship – A relation is created for the entity type. Add a FK within the same relation that references the PK of the relation. A recursive foreign key is a FK in a relation that references the PK values 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

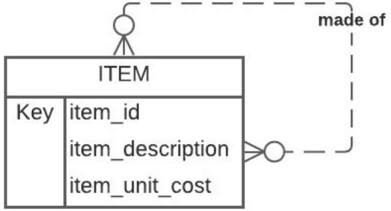


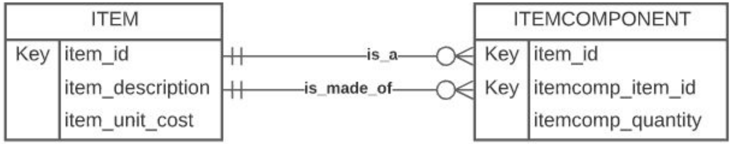


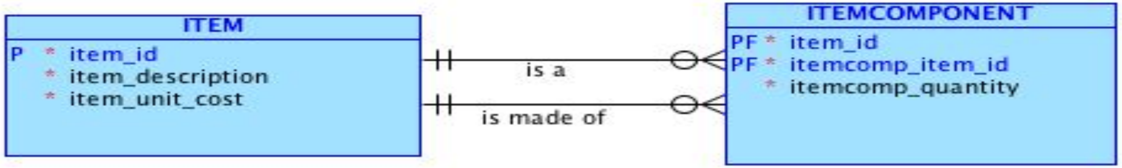
Mapping a M:N Unary Relationship



quantity?

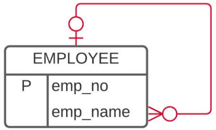


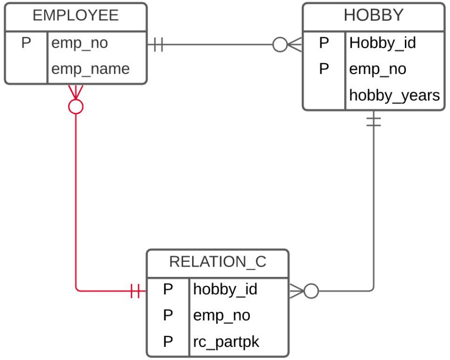




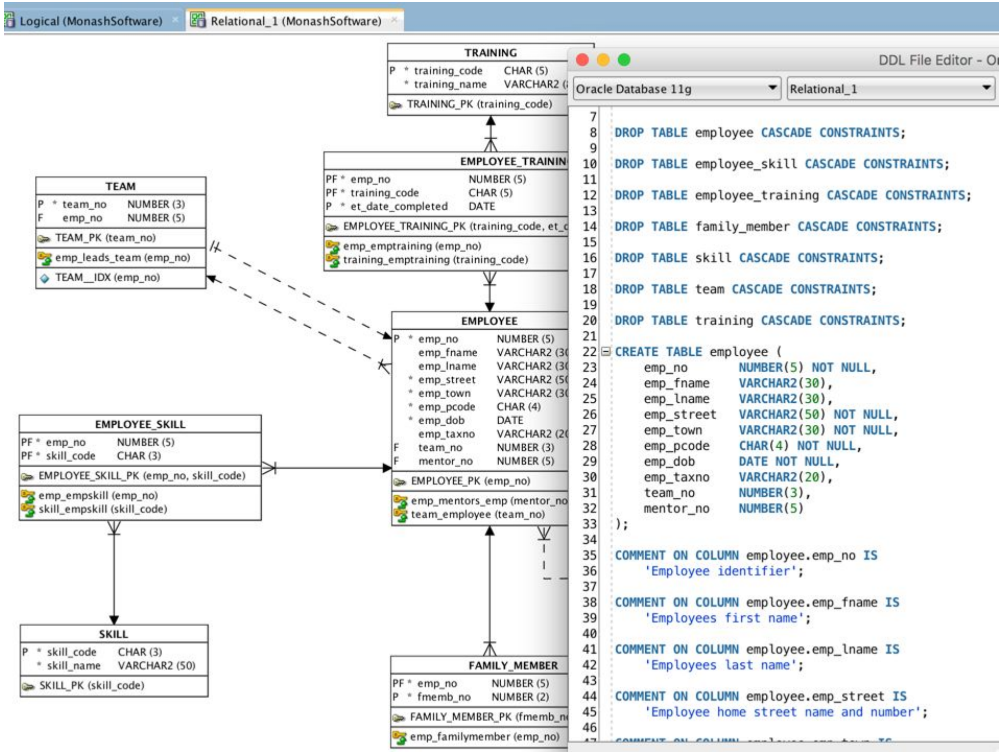
IMPORTANT NOTE

- 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:





SQL Developer 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_c
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)
```

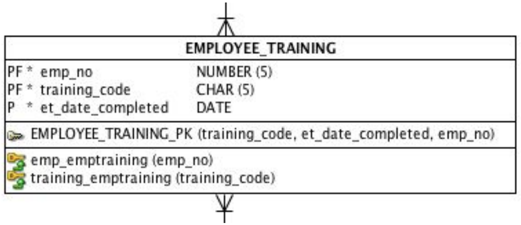
```
FAMILY_MEMBER
PF * emp_no        NUMBER (5)
P  * fmemb_no      NUMBER (2)
FAMILY_MEMBER_PK (fmemb_n(
emp_familymember (emp_no)
```

Oracle Database 11g

Relational_1

```

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

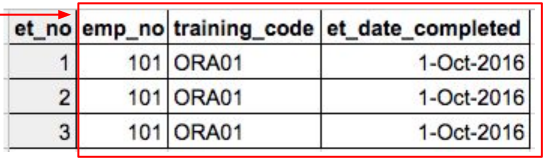


```
EMPLOYEE_TRAINING
PF * emp_no              NUMBER (5)
PF * training_code       CHAR (5)
P  * et_date_completed   DATE
EMPLOYEE_TRAINING_PK (training_code, et_date_completed, emp_no)
emp_emptraining (emp_no)
training_emptraining (training_code)
```

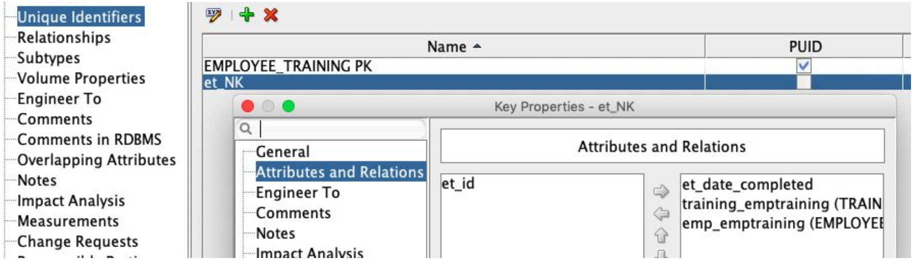
Potential problem:
Need to ensure that the identified key from the conceptual model - the natural key: (emp_no, training_code, et_date_completed) will still remain unique
Solution, where needed:
Define a unique index on the attributes of natural key

Surrogate PK's may be added **ONLY** on the logical model provided they are justified (include in documentation / assumptions).

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



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



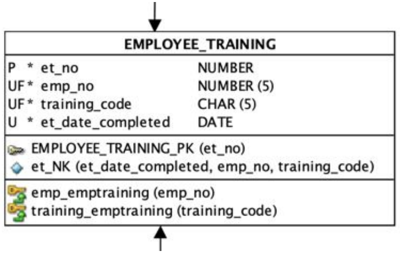
Name	PUID
EMPLOYEE_TRAINING PK	☑
et_NK	□

Key Properties - et_NK

Attributes and Relations

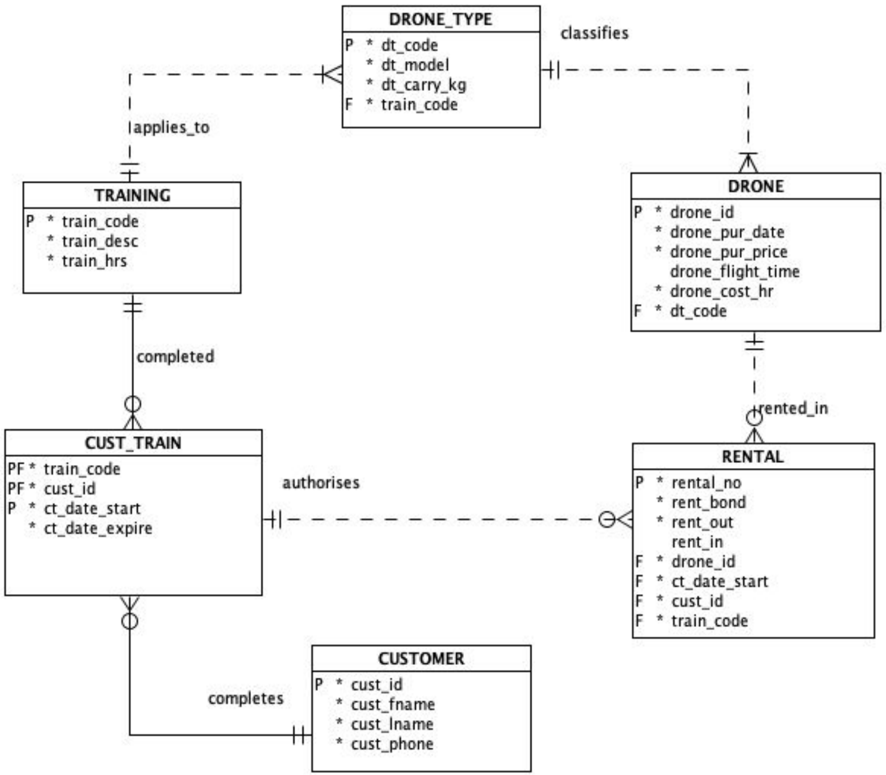
et_id

et_date_completed
training_emptraining (TRAIN
emp_emptraining (EMPLOYEE

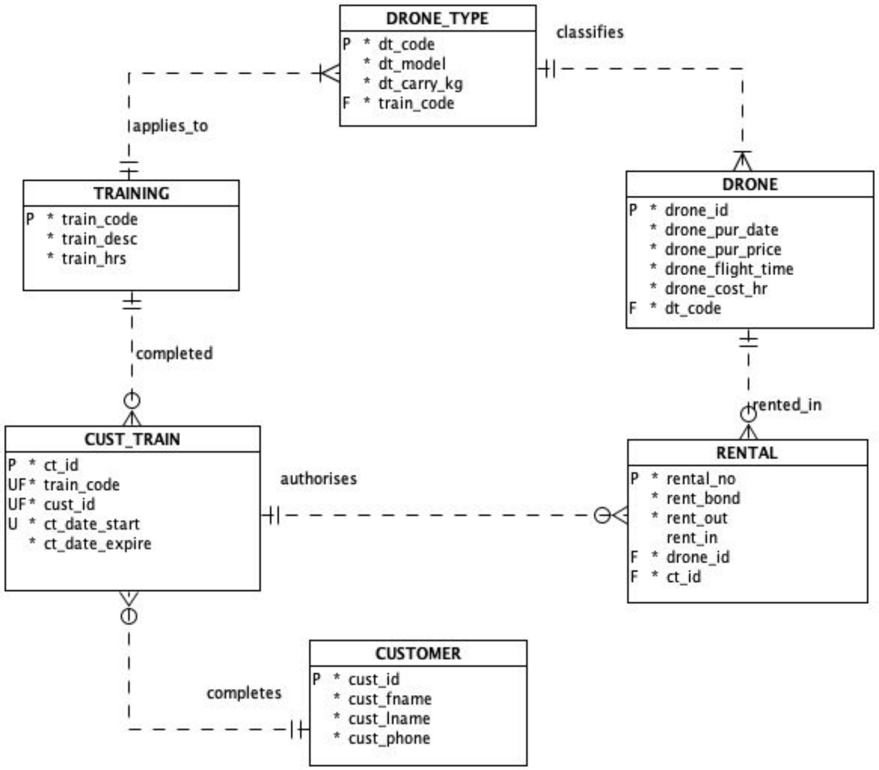


```
EMPLOYEE_TRAINING
P  * et_no               NUMBER
UF * emp_no              NUMBER (5)
UF * training_code       CHAR (5)
U  * et_date_completed   DATE
EMPLOYEE_TRAINING_PK (et_no)
et_NK (et_date_completed, emp_no, training_code)
emp_emptraining (emp_no)
training_emptraining (training_code)
```

Logical

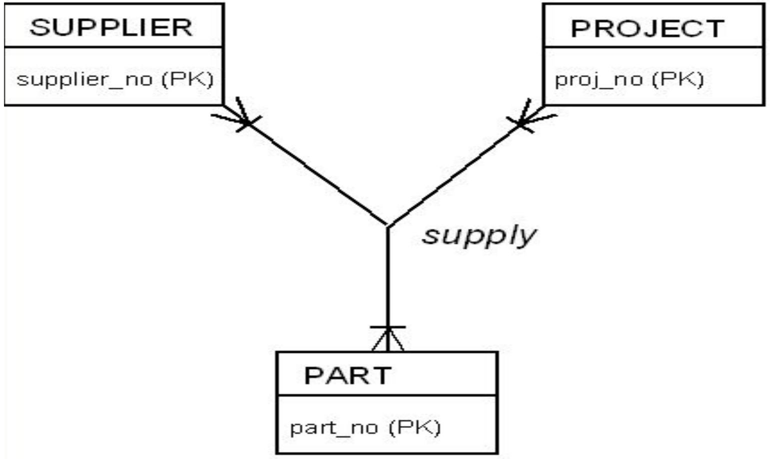


Logical - with Surrogate key

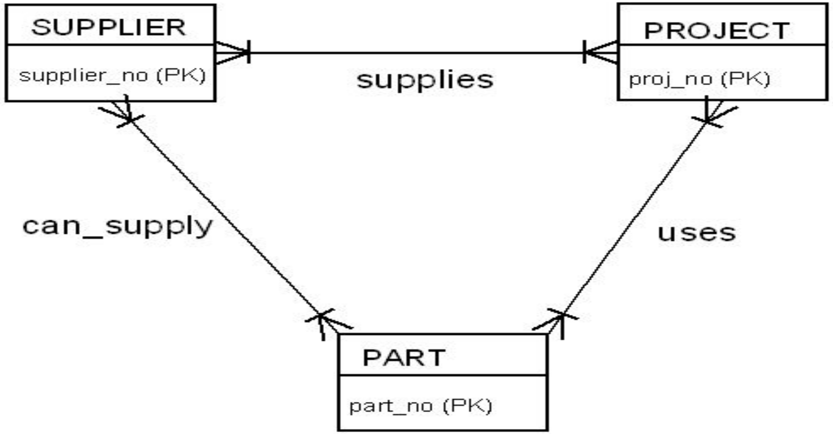


Ternary Relationships

Ternary



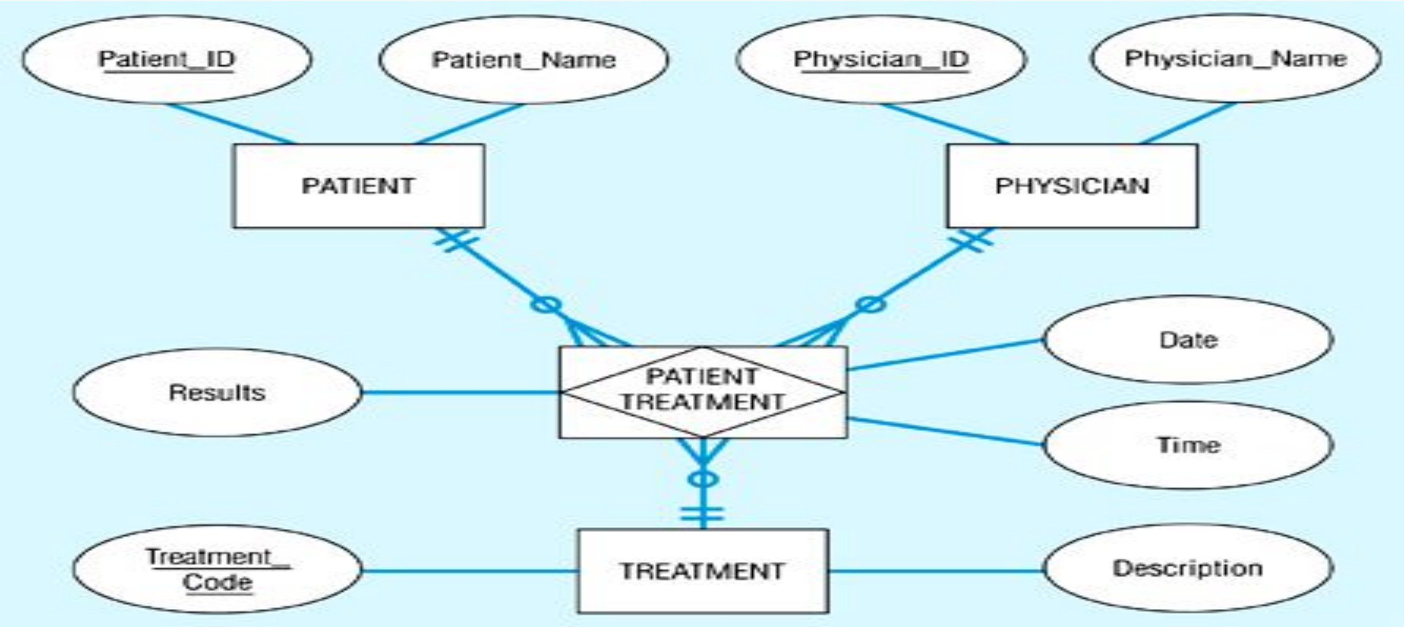
modelled as binary:



Ternary Relationships – model as binary relationships?

- Ternary represents more information than three binary relationships
- For example - Supplier 1 supplies Project 2 with Part 3 -
  - ternary
    - instance (supplier 1, project 2, part 3) exists
  - binaries
    - instances
      - (supplier1, project 2) (project 2, part 3) (supplier 1, part 3)
    - BUT does not imply (supplier 1, project 2, part 3)
- How then do we map such relationships?

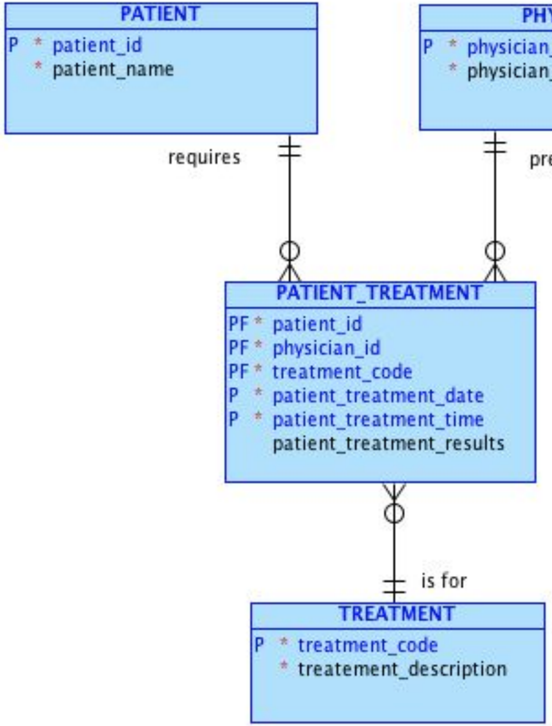
Mapping a Ternary Relationship



Map Ternary (and n-ary) Relationships

- Ternary relationship should be converted to an associative entity.
  - To map an associative entity type that links three regular entity types, an associative relation is created.
  - The default PK of this relation consists of the three PK attributes for the participating entity types.
  - Any attributes of the associative entity type become attributes of the new relation.

Mapping a Ternary Relationship

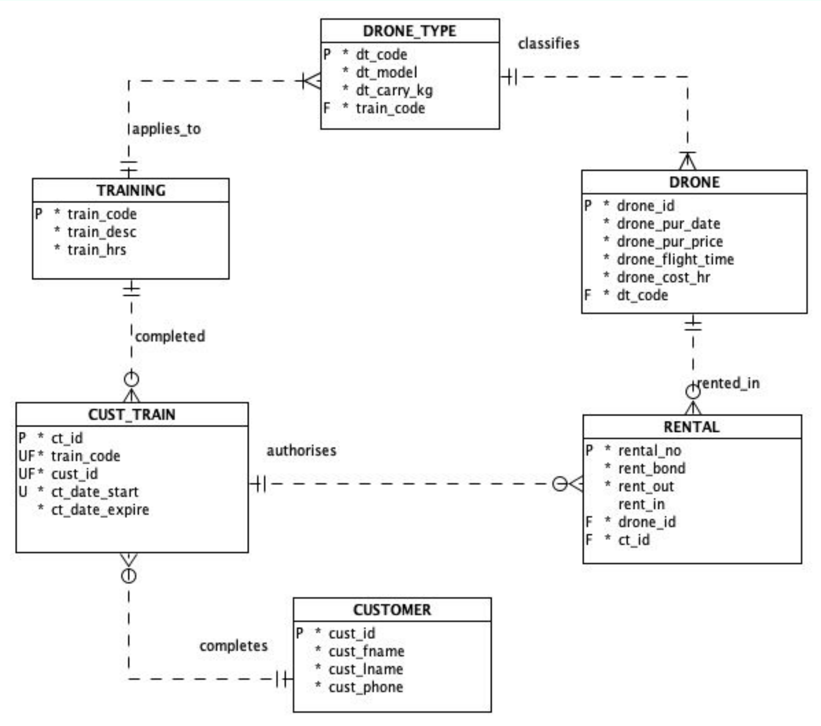


Overall Design Process - checklist

1. Assignment 1A
   a. complete conceptual model
2. Assignment 1B
   - free to modify submitted conceptual model in any manner - note will not be reassessed by marker (no requirement to submit again)

   a. normalise supplied forms to 3NF, one form at a time UNF->1NF->2NF->3NF
   b. carry out attribute synthesis on resultant set of 3NF relations to obtain one final set of 3NF relations
   c. map your 1A conceptual model to relational logical model
   d. integrate your final set of 3NF relations from 2b above, ensure attribute names in your normalisation are consistent with the names used in your logical model
   e. check model for no insert/update/delete anomalies
   f. check model for surrogate key requirement (if added ensure unique index on natural key created)
   g. generate physical (relational) model and from this generate the schema file, add appropriate details to the schema file (see week 6 tutorial)
   h. run the schema file and ensure no error (if there are any go back and fix logical model and repeat step 2g until removed)

Q4. What effect will the normalisation result have on the logical model?



3NF:

TRAINING (train_code, train_desc, train_active_mnths)

DRONE_TYPE (dt_code, dt_model, dt_manuf, train_code)

TRAINER (trainer_id, trainer_rego, trainer_fname, trainer_lname, trainer_category)

TRAINING_COURSE (train_code, traincourse_date, trainer_id)

CUSTOMER (cust_id, cust_fname, cust_lname)

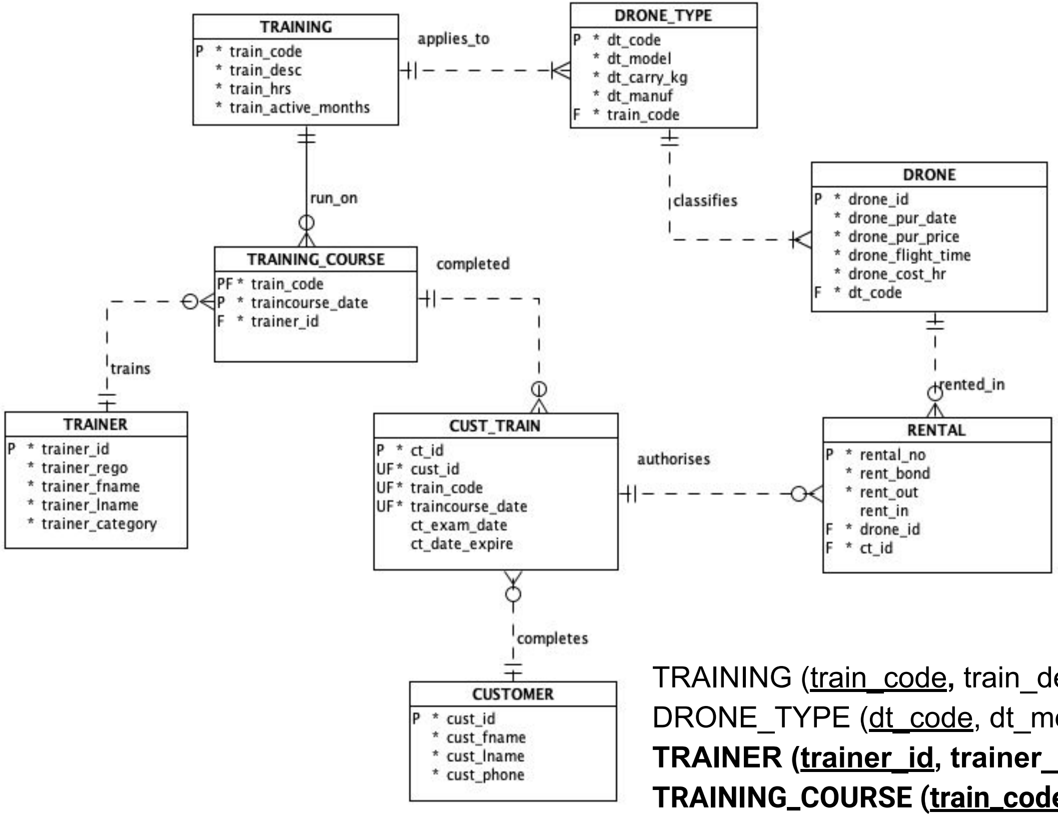
CUST_TRAINING (train_code, traincourse_date, cust_id, ct_exam_date, ct_date_expire)

A. Change the attributes of TRAINING
B. Change the attributes of CUSTOMER
C. Change the PK in TRAINING
D. Add a new relation TRAINER
E. Add a new relation CUST_TRAINING

Multiple answers possible

Q5. Name an attribute on this model which exhibits insert/update/delete anomalies

- **Free text (word cloud) response**



TRAINING
P * train_code
* train_desc
* train_hrs
* train_active_months

DRONE_TYPE
P * dt_code
* dt_model
* dt_carry_kg
* dt_manuf
F * train_code

DRONE
P * drone_id
* drone_pur_date
* drone_pur_price
* drone_flight_time
* drone_cost_hr
F * dt_code

TRAINING_COURSE
PF * train_code
P * traincourse_date
F * trainer_id

TRAINER
P * trainer_id
* trainer_rego
* trainer_fname
* trainer_lname
* trainer_category

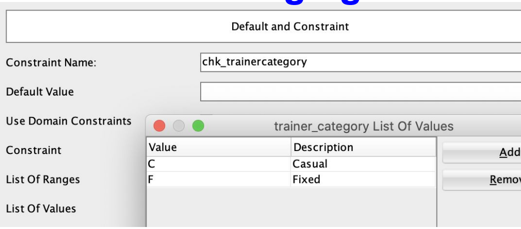
CUST_TRAIN
P * ct_id
UF * cust_id
UF * train_code
UF * traincourse_date
ct_exam_date
ct_date_expire

RENTAL
P * rental_no
* rent_bond
* rent_out
rent_in
F * drone_id
F * ct_id

CUSTOMER
P * cust_id
* cust_fname
* cust_lname
* cust_phone

TRAINING (train_code**,** train_desc, **train_active_mnths**)
DRONE_TYPE (dt_code, dt_model, **dt_manuf**, train_code)
TRAINER (trainer_id, trainer_rego, trainer_fname, trainer_lname, trainer_category)
TRAINING_COURSE (train_code, traincourse_date, trainer_id)
CUSTOMER (cust_id, cust_fname, cust_lname)
CUST_TRAINING (train_code**, traincourse_date,** cust_id, ct_exam_date, ct_date_expire)

check constraint
- small unchanging set => codify



Default and Constraint

Constraint Name: chk_trainercategory

Default Value

Use Domain Constraints

Constraint

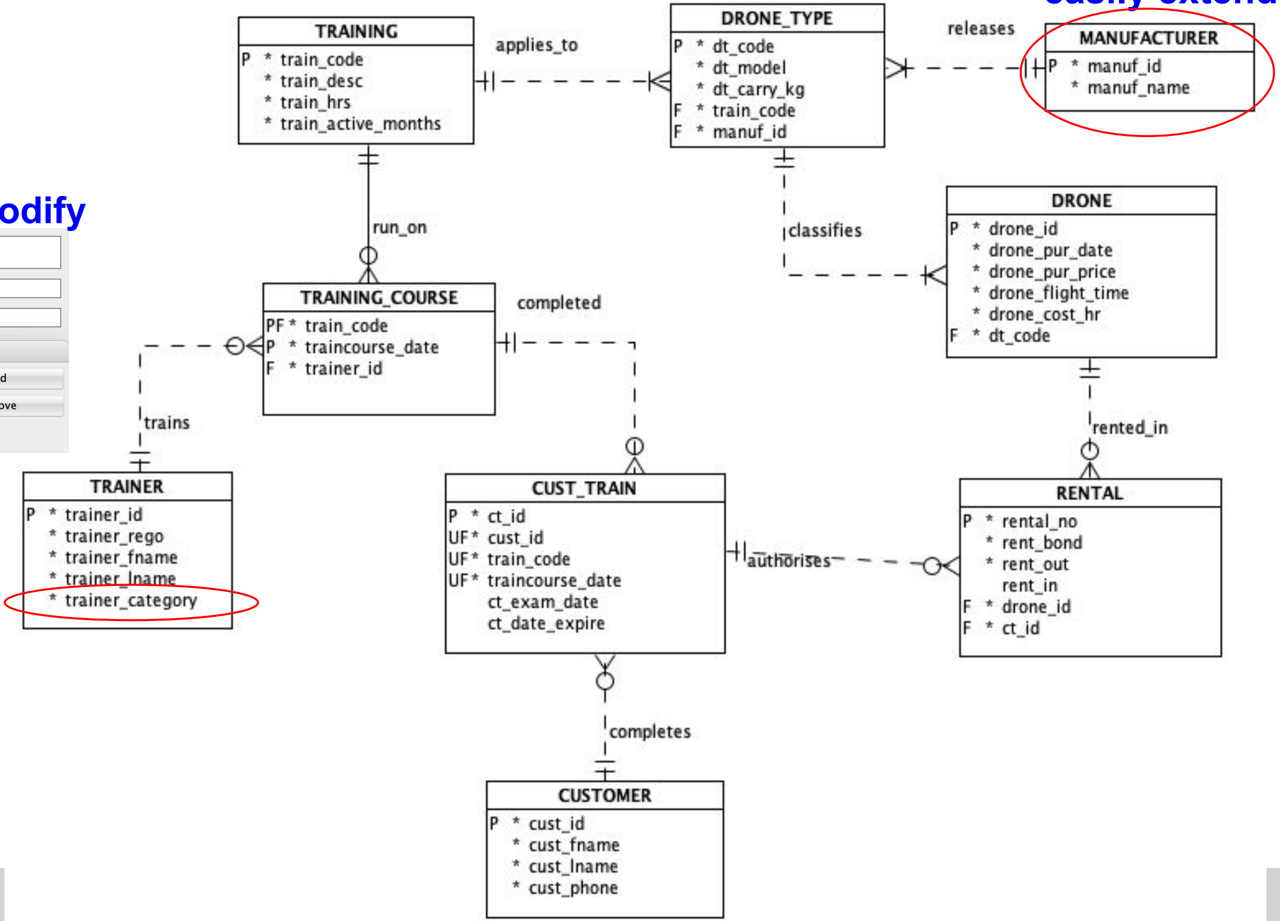
List Of Ranges

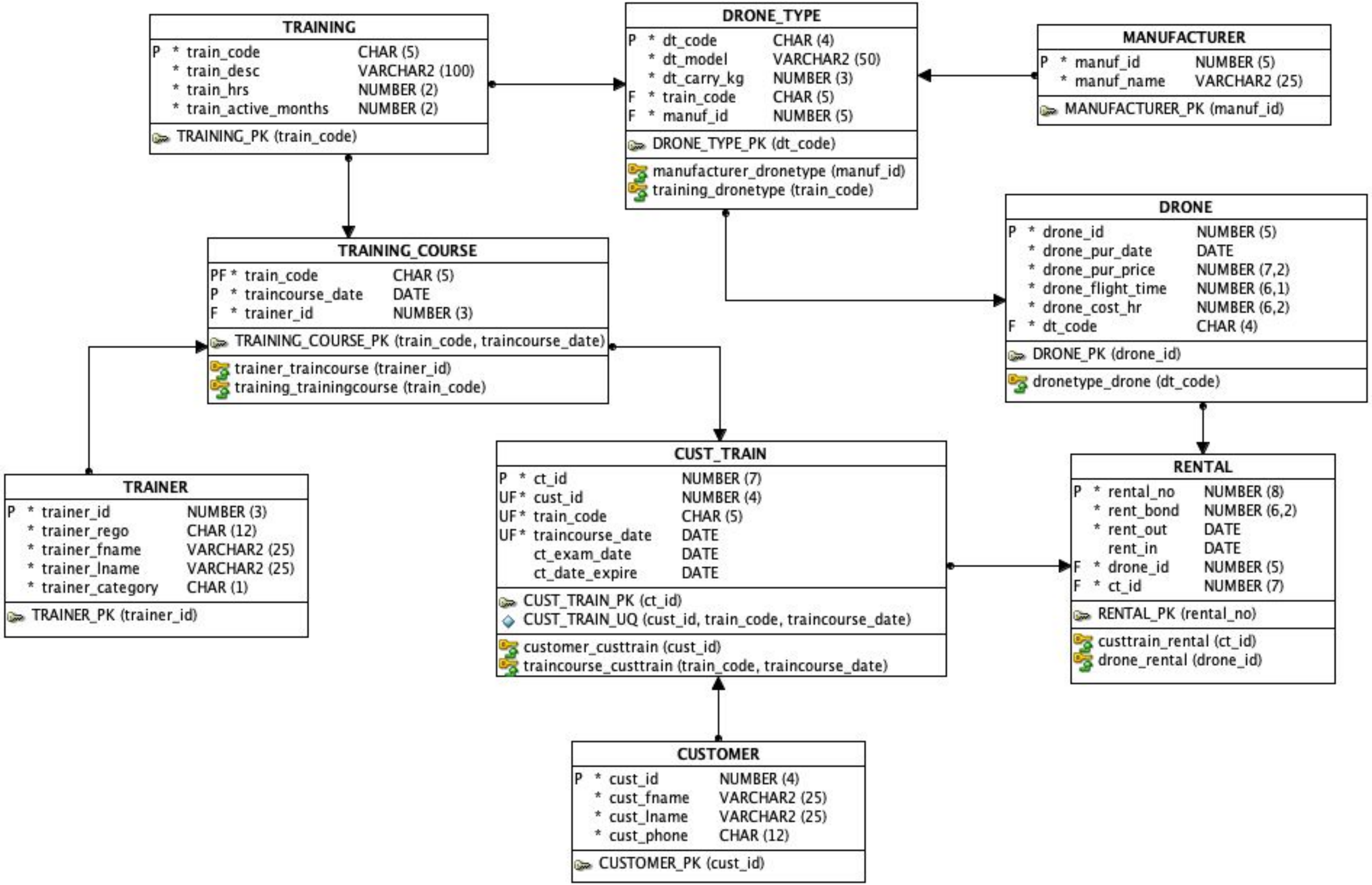
List Of Values

trainer_category List Of Values

Value	Description
C	Casual
F	Fixed

lookup table
- easily extended





TRAINING

P *	train_code	CHAR (5)
*	train_desc	VARCHAR2 (100)
*	train_hrs	NUMBER (2)
*	train_active_months	NUMBER (2)

TRAINING_PK (train_code)

DRONE_TYPE

P *	dt_code	CHAR (4)
*	dt_model	VARCHAR2 (50)
*	dt_carry_kg	NUMBER (3)
F *	train_code	CHAR (5)
F *	manuf_id	NUMBER (5)

DRONE_TYPE_PK (dt_code)

manufacturer_dronetype (manuf_id)
training_dronetype (train_code)

MANUFACTURER

P *	manuf_id	NUMBER (5)
*	manuf_name	VARCHAR2 (25)

MANUFACTURER_PK (manuf_id)

TRAINING_COURSE

PF *	train_code	CHAR (5)
P *	traincourse_date	DATE
F *	trainer_id	NUMBER (3)

TRAINING_COURSE_PK (train_code, traincourse_date)

trainer_traincourse (trainer_id)
training_trainingcourse (train_code)

DRONE

P *	drone_id	NUMBER (5)
*	drone_pur_date	DATE
*	drone_pur_price	NUMBER (7,2)
*	drone_flight_time	NUMBER (6,1)
*	drone_cost_hr	NUMBER (6,2)
F *	dt_code	CHAR (4)

DRONE_PK (drone_id)

dronetype_drone (dt_code)

TRAINER

P *	trainer_id	NUMBER (3)
*	trainer_rego	CHAR (12)
*	trainer_fname	VARCHAR2 (25)
*	trainer_lname	VARCHAR2 (25)
*	trainer_category	CHAR (1)

TRAINER_PK (trainer_id)

CUST_TRAIN

P *	ct_id	NUMBER (7)
UF *	cust_id	NUMBER (4)
UF *	train_code	CHAR (5)
UF *	traincourse_date	DATE
	ct_exam_date	DATE
	ct_date_expire	DATE

CUST_TRAIN_PK (ct_id)
◇ CUST_TRAIN_UQ (cust_id, train_code, traincourse_date)

customer_custtrain (cust_id)
traincourse_custtrain (train_code, traincourse_date)

RENTAL

P *	rental_no	NUMBER (8)
*	rent_bond	NUMBER (6,2)
*	rent_out	DATE
	rent_in	DATE
F *	drone_id	NUMBER (5)
F *	ct_id	NUMBER (7)

RENTAL_PK (rental_no)

custtrain_rental (ct_id)
drone_rental (drone_id)

CUSTOMER

P *	cust_id	NUMBER (4)
*	cust_fname	VARCHAR2 (25)
*	cust_lname	VARCHAR2 (25)
*	cust_phone	CHAR (12)

CUSTOMER_PK (cust_id)