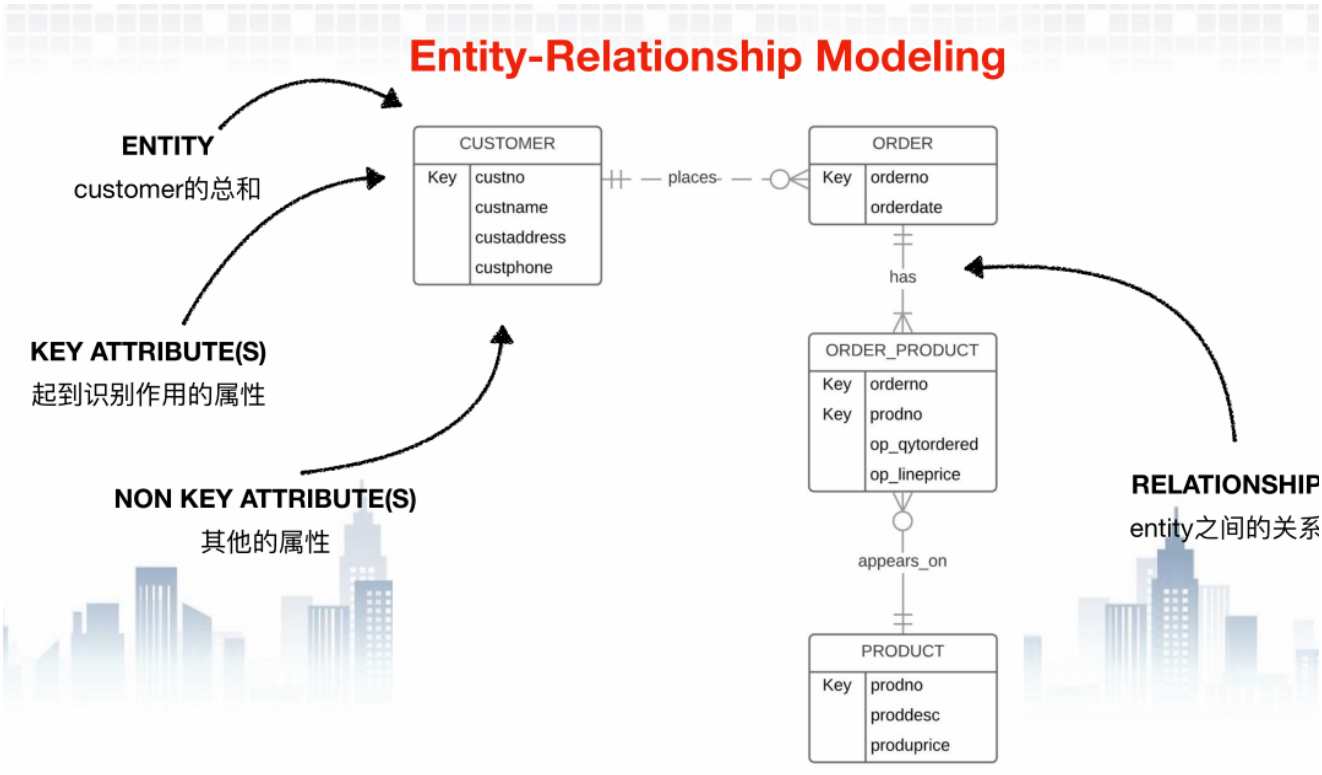
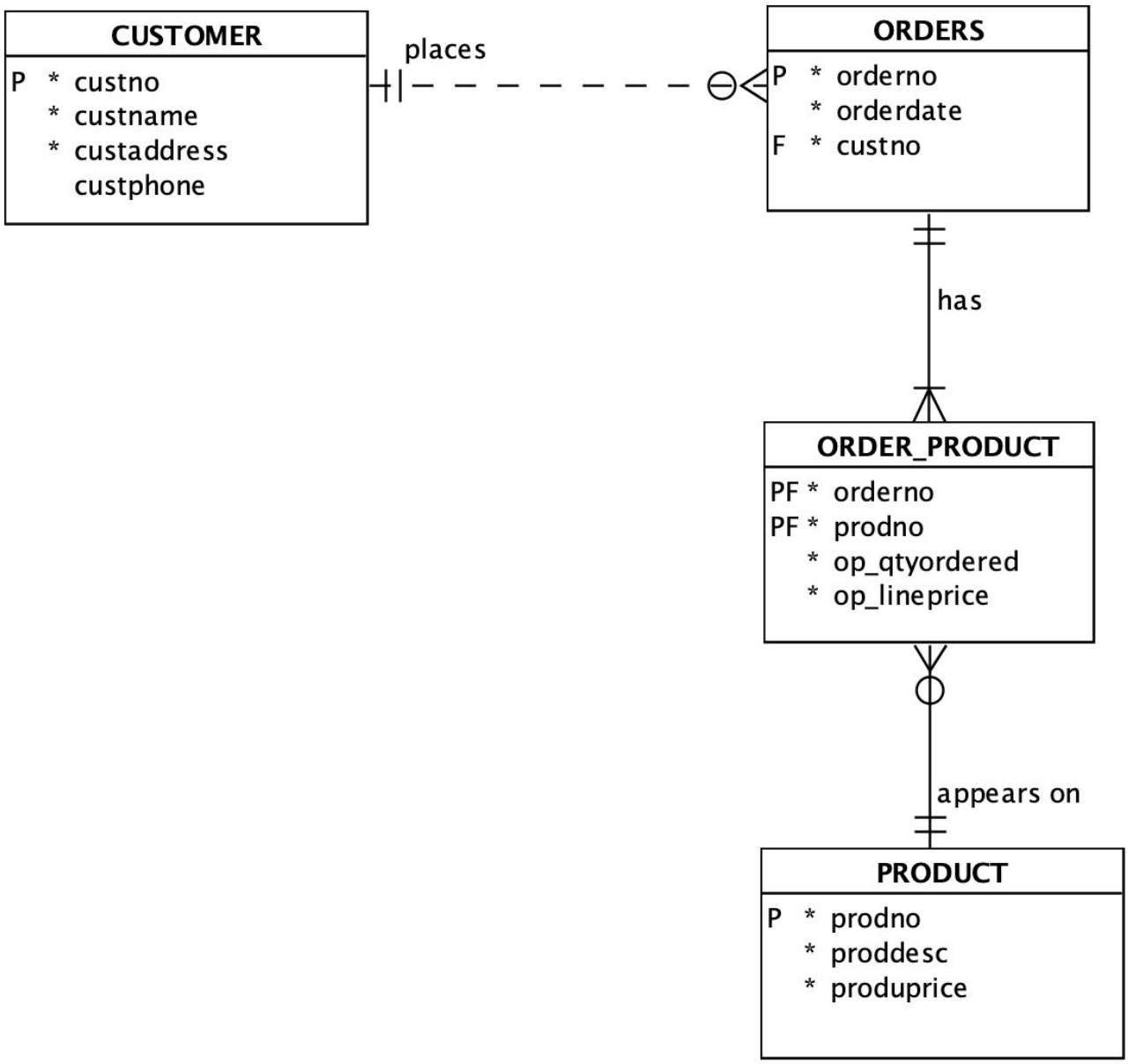
WEEK 1

ER Modeling



Logical Model



Physical Model

Oracle Database 12c | Relational_1 | Generate

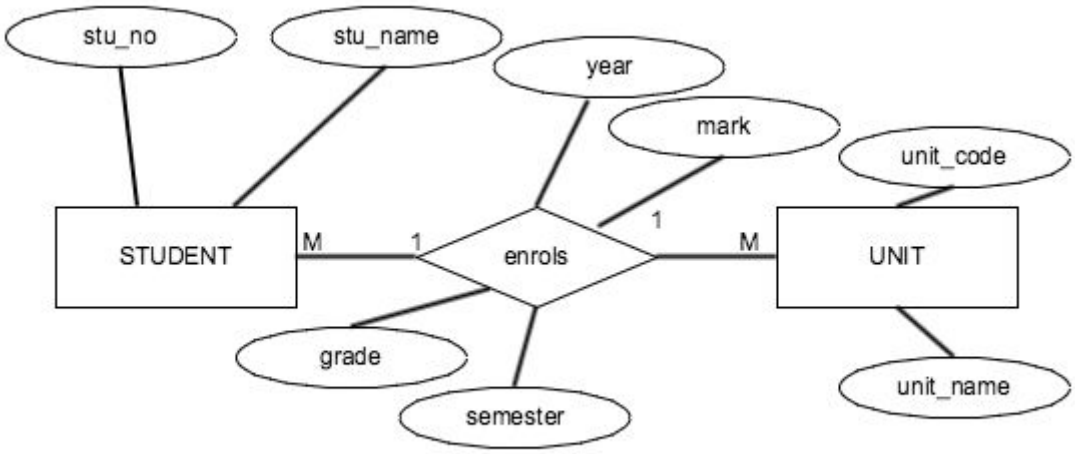
```
 8  CREATE TABLE customer (
 9      custno        NUMBER(7) NOT NULL,
10      custname      VARCHAR2(50) NOT NULL,
11      custaddress   VARCHAR2(50) NOT NULL,
12      custphone     CHAR(10)
13  );
14
15  COMMENT ON COLUMN customer.custno IS
16      'Customer number';
17
18  COMMENT ON COLUMN customer.custname IS
19      'Customer name';
20
21  COMMENT ON COLUMN customer.custaddress IS
22      'Customer address';
23
24  COMMENT ON COLUMN customer.custphone IS
25      'Customer phone number';
26
27  ALTER TABLE customer ADD CONSTRAINT customer_pk PRIMARY KEY ( custno );
28
29  CREATE TABLE order_product (
30      orderno         NUMBER(7) NOT NULL,
31      prodno          NUMBER(7) NOT NULL,
32      op_qtyordered   NUMBER(3) NOT NULL,
33      op_lineprice    NUMBER(8, 2) NOT NULL
34  );
35
```

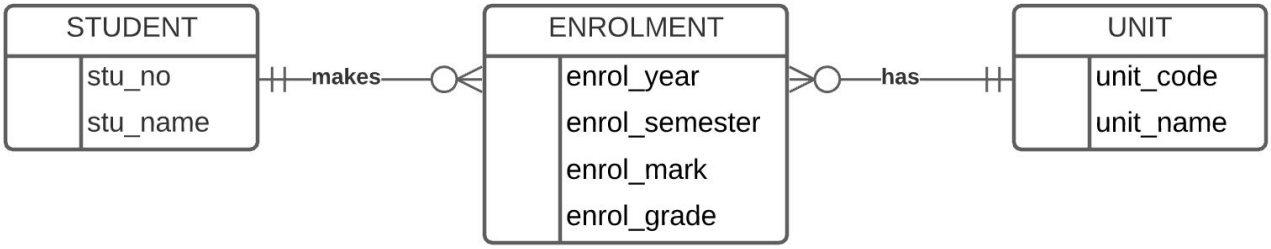
week 2 Conceptual Modelling

理解数据库设计的各个阶段
理解ANSI/SPARC结构在数据库设计中的角色
认识实体关系图当中的各个组成部分
理解strong/weak entity之间的不同
学习绘制概念模型图

ER diagram

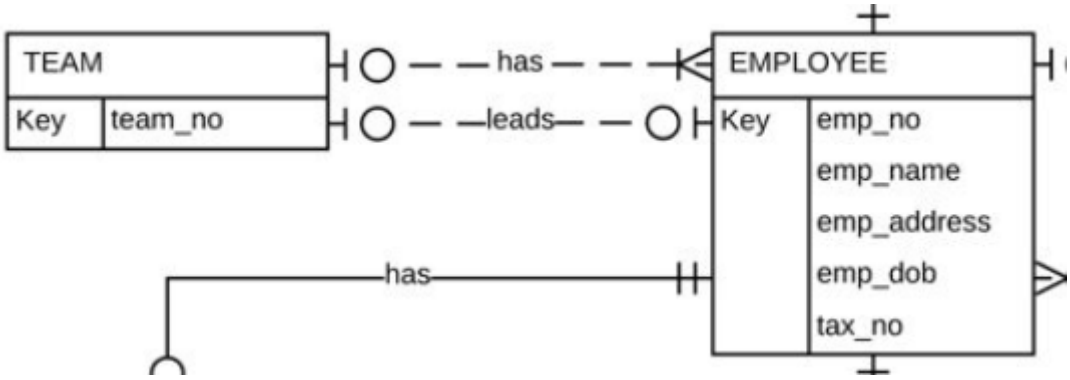
Entities





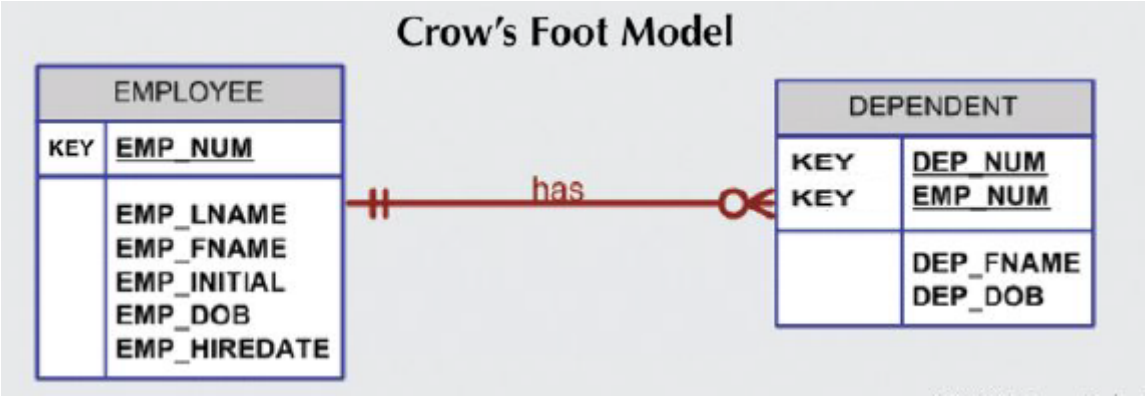
Strong Entity

Has a key which may be defined without reference to other entities



Weak Entity

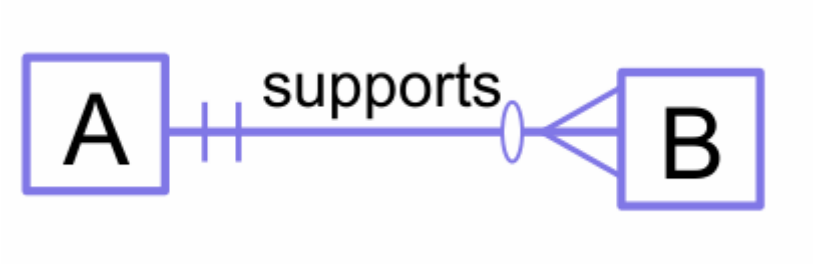
Has a key which requires the existence of one or more other entities.
For example FAMILY entity - need to include the key of employee to create a suitable key for family



Relationship

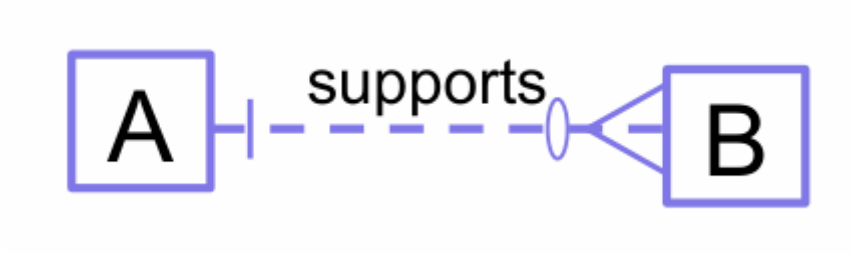
Identifying

- Identifier of A is part of identifier of B
- Shown with solid line
- ENROLMENT - STUDENT Enrolment key includes student id,which is an identifier of student.



Non-identifying

- Identifier of A is NOT part of identifier of B.
- Shown with broken line
- Department no (identifier of department) is not part of Employee's identifier



Attributes

Simple

Cannot be subdivided
Age, sex, marital status

Composite

Can be subdivided into additional attributes
Address into street, city

Single-valued

Can have only a single value
Person has one social security number

Multi-valued

Can have many values
Person may have several college degrees

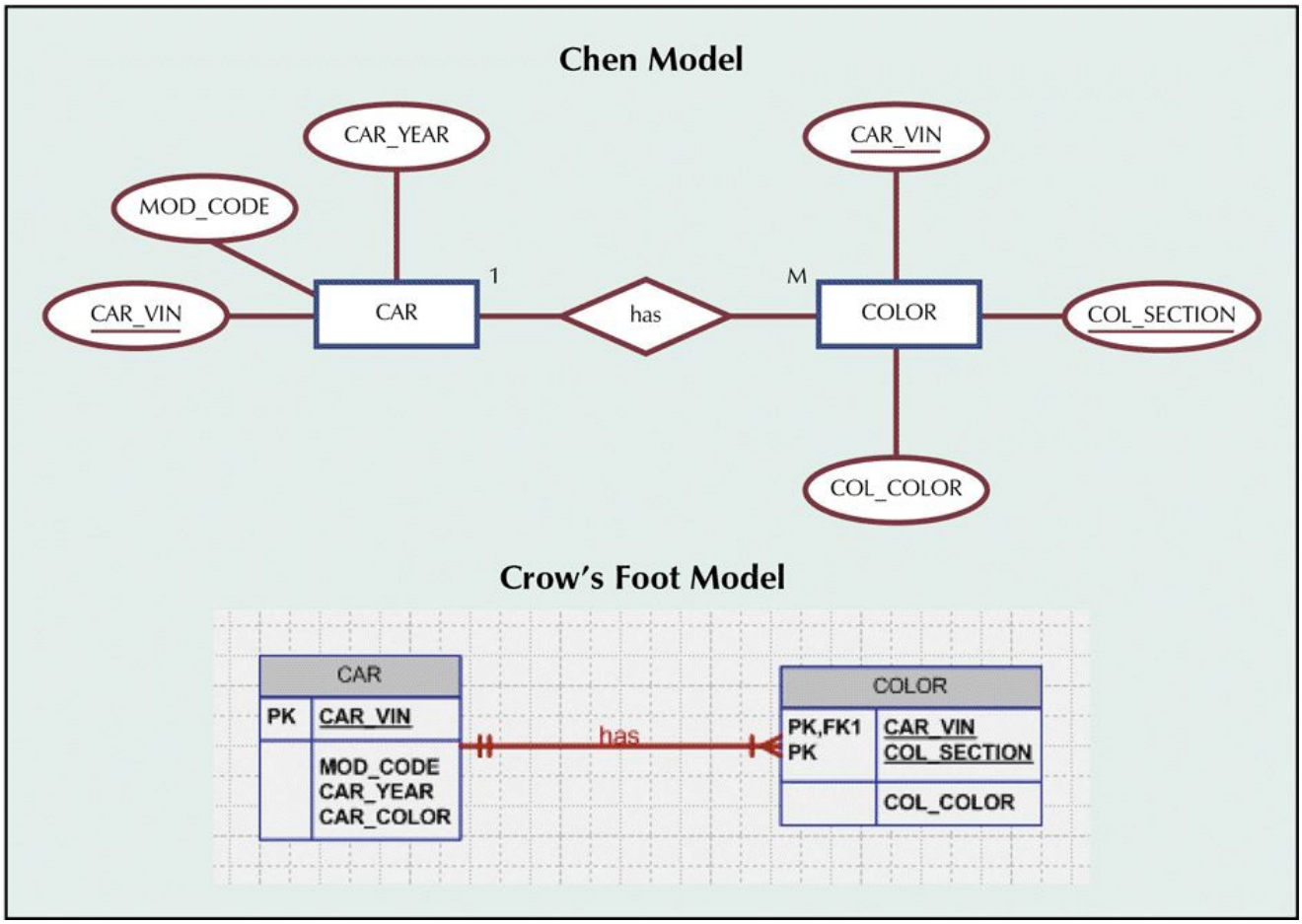
Derived

Can be derived with algorithm
Age can be derived from date of birth

Multivalued Attribute

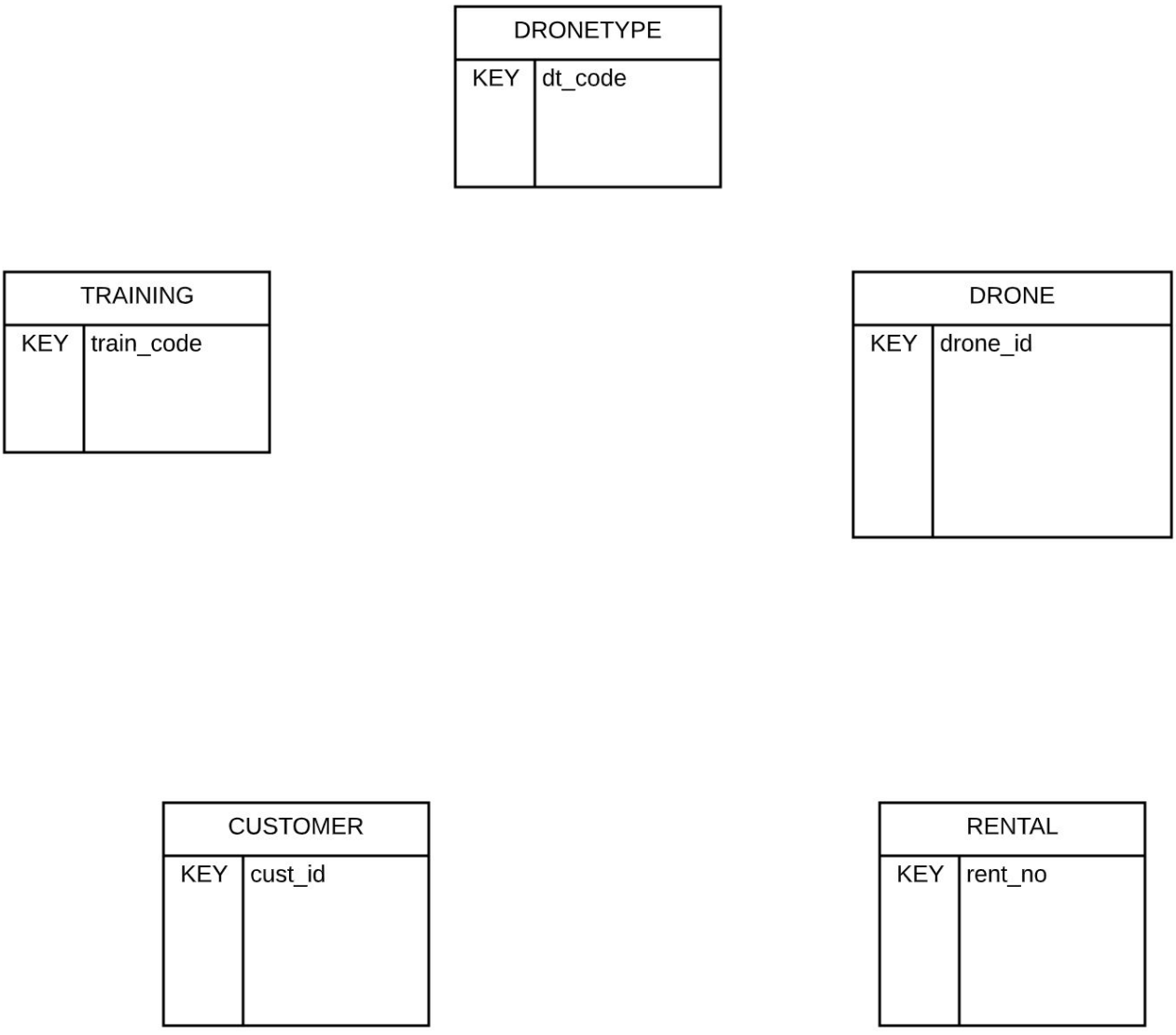
An attribute that has a list of values.
Car colour may consist of body colour, trim colour, bumper colour.

FIGURE 4.5 A NEW ENTITY SET COMPOSED OF A MULTIVALUED ATTRIBUTE'S COMPONENTS



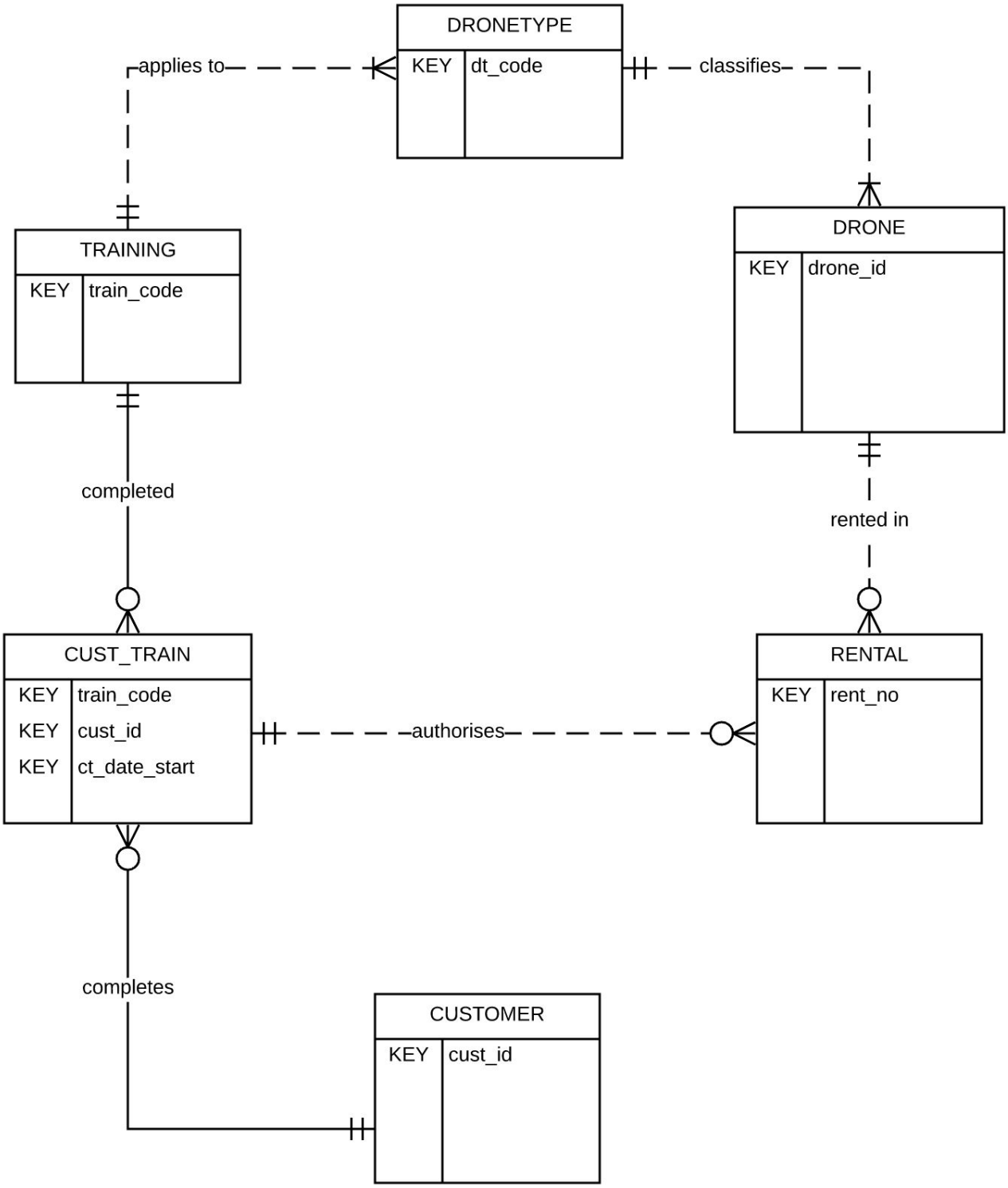
ERD

Step 1 Identify Main Entities

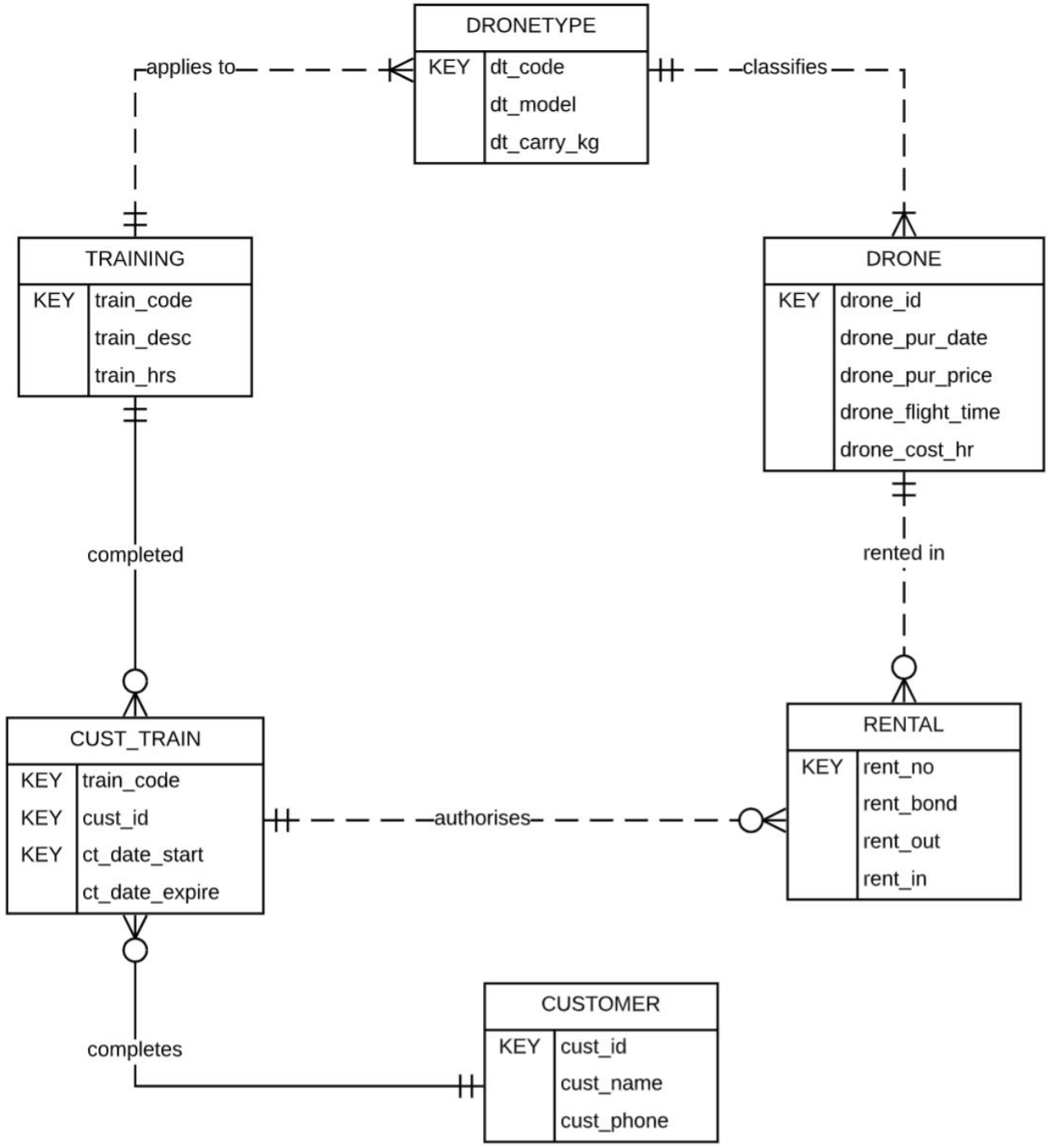


STEP 2: Identify the relationships which exist between these entities

(remember to add an appropriate verb)



Step 3 Add Non-Key Attributes



Exercise answer

WEEK 3 Relational Database Model

• Explain the relational model and its components
• Identify and evaluate the issues of insert, update and delete anomalies
• Define entity and referential integrity
• Differentiate the types of keys and distinguish their roles in the relational model
• Formulate and write relational algebra to solve query scenarios

Basic structure is the mathematical concept of a RELATION mapped to the 'concept' of a table (tabular representation of relation)
– Relation - abstract object
– Table - pictorial representation
– Storage structure - "real thing" - eg. isam file of 1's and 0's

A Relation

A relation consists of two parts
– heading – body
Relation heading ⇨ column headings
Relation body ⇨ set of data rows

Relation Heading ⇨ column headings

Also called Relational Schema consists of a fixed set of attributes

`R (A1,A2,.....An)` R = relation name, Ai = attribute i`

Each attribute corresponds to one underlying domain:
Customer relation heading:
CUSTOMER (custno, custname, custadd, custcredlimit)
dom(custno) = customer_number

Relation Body ⇨ set of data rows

Relation Properties

No duplicate tuples

sets do not contain duplicate elements hence tuples must be unique

Tuples are unordered within a relation

sets are not ordered hence tuples can only be accessed by content

No ordering of attributes within a tuple

Tuple values are atomic - cannot be divided

`EMPLOYEE (eid, ename, departno, dependants)`
not allowed: `dependants (dep_name, dep_age)` multivalued hence no multivalued (repeating) attributes allowed, called the first normal form rule

Functional Dependency

A set of attributes A functionally determines an attribute B if, and only if, for each A value, there is exactly one value of B in the relation. It is denoted as

A → B (A determines B, or B depends on A)

Relational Model Keys

superkey

A superkey of a relation R is an attribute or set of attributes which exhibits only the uniqueness property
– No two tuples of R have the same value for the superkey (Uniqueness property)
– t1[superkey] ≠ t2[superkey]

candidate key (CK)

A candidate key (CK) of a relation R is an attribute or set of attributes which exhibits the following properties:
– Uniqueness property (as above), and
– No proper subset of CK has the uniqueness property (Minimality or Irreducibility property) ie. a minimal superkey
–One candidate key is chosen to be the primary key (PK) of a relation

Primary key

One candidate key is chosen to be the primary key (PK) of a relation
preferably a single attribute, preferably numeric

TABLE 5.3

DESIRABLE PRIMARY KEY CHARACTERISTICS

PK CHARACTERISTIC	RATIONALE
Unique values	The PK must uniquely identify each entity instance. A primary key must be able to guarantee unique values. It cannot contain nulls.
Nonintelligent	The PK should not have embedded semantic meaning other than to uniquely identify each entity instance. An attribute with embedded semantic meaning is probably better used as a descriptive characteristic of the entity than as an identifier. For example, a student ID of 650973 would be preferred over Smith, Martha L. as a primary key identifier.
No change over time	If an attribute has semantic meaning, it might be subject to updates, which is why names do not make good primary keys. If Vickie Smith is the primary key, what happens if she changes her name when she gets married? If a primary key is subject to change, the foreign key values must be updated, thus adding to the database work load. Furthermore, changing a primary key value means that you are basically changing the identity of an entity. In short, the PK should be permanent and unchangeable.
Preferably single-attribute	A primary key should have the minimum number of attributes possible (irreducible). Single-attribute primary keys are desirable but not required. Single-attribute primary keys simplify the implementation of foreign keys. Having multiple-attribute primary keys can cause primary keys of related entities to grow through the possible addition of many attributes, thus adding to the database workload and making (application) coding more cumbersome.
Preferably numeric	Unique values can be better managed when they are numeric, because the database can use internal routines to implement a counter-style attribute that automatically increments values with the addition of each new row. In fact, most database systems include the ability to use special constructs, such as Autonumber in Microsoft Access, sequence in Oracle, or uniqueidentifier in MS SQL Server to support self-incrementing primary key attributes.
Security-compliant	The selected primary key must not be composed of any attribute(s) that might be considered a security risk or violation. For example, using a Social Security number as a PK in an EMPLOYEE table is not a good idea.

Null

NULL is NOT a value - is a representation of the fact that there is NO VALUE
VALUE NOT APPLICABLE
VALUE UNKNOWN
VALUE DOES NOT EXIST
VALUE UNDEFINED

Writing Relations

Relations may be represented using the following notation:
– RELATION_NAME (attribute1, attribute2,...)
▪ Relation_name must not be pluralised (is a set name)
▪ The primary key is underlined.

```
STAFF (staff_id, staff_surname, staff_initials, staff_address, staff_phone)
```

Foreign Key (FK)

An attribute/s in a relation that exists in the same, or another relation as a Primary Key

Data Integrity

Entity integrity
Primary key value must not be NULL.
• No duplicate tuple property then ensures that each primary key must be unique
• Implemented in the RDBMS via a unique index on the PK

Referential integrity
– The values of FK must either match a value of a full PK in the related relation or be NULL.

Column/Domain integrity
– All values in a given column must come from the same domain (the same data type and range)

Relational Algebra

8 basic operations:
– single relation:
• selection,• projection

– two relations:
• cartesian product, join • union
• intersection • difference • division

PROJECT π

PRDETAIL (project_code, project_manager, project_bid_price)

$$\pi$$

PROJECT_CODE	PROJECT_MANAGER	PROJECT_BID_PRICE
21-5Z	Holly B. Parker	$16,833,460.00
25-2D	Jane D. Grant	$12,500,000.00
25-5A	George F. Dorts	$32,512,420.00
25-9T	Holly B. Parker	$21,563,234.00
27-4Q	George F. Dorts	$10,314,545.00
29-2D	Holly B. Parker	$25,559,999.00
31-7P	William K. Moor	$56,850,000.00

$$\text{RESULT} = \pi_{\text{project\_manager}} \text{PRDETAIL}$$

SELECT σ

PRDETAIL (project_code, project_manager, project_bid_price)

$$\sigma$$

PROJECT_CODE	PROJECT_MANAGER	PROJECT_BID_PRICE
21-5Z	Holly B. Parker	$16,833,460.00
25-2D	Jane D. Grant	$12,500,000.00
25-5A	George F. Dorts	$32,512,420.00
25-9T	Holly B. Parker	$21,563,234.00
27-4Q	George F. Dorts	$10,314,545.00
29-2D	Holly B. Parker	$25,559,999.00
31-7P	William K. Moor	$56,850,000.00

$$\text{RESULT} = \sigma_{\text{project\_code} = 25\text{-}5A} \text{PRDETAIL}$$

Multiple Actions

PRDETAIL (project_code, project_manager, project_bid_price)

2

PROJECT_CODE	PROJECT_MANAGER	PROJECT_BID_PRICE
21-5Z	Holly B. Parker	$16,833,460.00
25-2D	Jane D. Grant	$12,500,000.00
25-5A	George F. Dorts	$32,512,420.00
25-9T	Holly B. Parker	$21,563,234.00
27-4Q	George F. Dorts	$10,314,545.00
29-2D	Holly B. Parker	$25,559,999.00
31-7P	William K. Moor	$56,850,000.00

1

$$RESULT = \pi_{project\_manager} (\sigma_{project\_code = 25\text{-}5A} PRDETAIL)$$

NATURAL JOIN

Step 1: STUDENT X MARK

STUDENT

ID	Name
1	Alice
2	Bob

MARK

ID	Subj	Marks
1	1004	95
2	1045	55
1	1045	90

Step 2: delete rows where IDs do not match (select =)

STUDENT.ID	Name	MARK.ID	Subj	Marks
1	Alice	1	1004	95
~~1~~	~~Alice~~	~~2~~	~~1045~~	~~55~~
1	Alice	1	1045	90
~~2~~	~~Bob~~	~~1~~	~~1004~~	~~95~~
2	Bob	2	1045	55
~~2~~	~~Bob~~	~~1~~	~~1045~~	~~90~~

Step 3: delete duplicate columns (project away)

ID	Name	Subj	Marks
1	Alice	1004	95
1	Alice	1045	90
2	Bob	1045	55

A natural join of STUDENT and MARK

UNION, INTERSECT, DIFFERENCE

UNION, INTERSECT, DIFFERENCE

STOREA

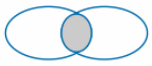
product_id	product_name
1	LG Nano91 75" 4K
2	TCL P725 65" 4K UHD
3	Sony X85J 75" Bravia

UNION (STOREA ∪ STOREB)

product_id	product_name
1	LG Nano91 75" 4K
2	TCL P725 65" 4K UHD
3	Sony X85J 75" Bravia
33	LG C1 48" Self Lit OLED 4K

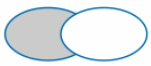
STOREB

product_id	product_name
1	LG Nano91 75" 4K
2	TCL P725 65" 4K UHD
33	LG C1 48" Self Lit OLED 4K



INTERSECT (STOREA ∩ STOREB)

product_id	product_name
1	LG Nano91 75" 4K
2	TCL P725 65" 4K UHD



DIFFERENCE (STOREA - STOREB)

product_id	product_name
3	Sony X85J 75" Bravia

Union compatible relations required

quiz and exercise

Q7. Relational Algebra

The following relations represent a karate dojo member training attendance:

SENSEI (sensei_id, sensei_name)
TRAINING_SCHEDULE (training_day, training_time, group_id, sensei_id)
ATTENDANCE (training_day, training_time, member_id, attendance_date)
MEMBER (member_id, member_name, member_dob, member_belt, group_id)
GROUP (group_id, group_name, group_age_range)

A. Primary keys are underlined
B. A karate member falls into one of the age level groups: Tiny Tiger (for 4-7 year old), Young Dragon (for 8-14 years old), or Adult (for 14+ years old) and owns a certain color of belt (e.g. white, green, brown or black)
C. Sensei (Karate teachers) are scheduled to train an age level group of karate members in a particular day and time (e.g. Sensei Luke Nakamura trains Tiny Tiger members every Tuesday 5pm)
D. A karate member may attend more than one training schedule of their age level group in a given week.

Write the relational algebra for the following query (**your answer must show an understanding of query efficiency**):

(2) Show the name, belt colour and attendance dates of the member with an id of 12345

ANSWER Q7

(2) Show the name, belt colour and attendance dates of the member with an id of 12345

$$R2 = \pi\ member\_name, member\_belt, attendance\_date\ (\sigma\ member\_id = 12345\ (\ MEMBER \bowtie ATTENDANCE))$$

- this is the CANONICAL QUERY - not technically incorrect, but very inefficient, say member 12345 has only attended once in say 1000 tuples in ATTENDANCE. The join between MEMBER and ATTENDANCE yields, in such a scenario, 1000 tuples, 999 of which are unnecessary.

Your solution must demonstrate an understanding of efficiency:

$$A2a = \pi\ member\_id, attendance\_date\ (\sigma\ member\_id = 12345\ ATTENDANCE)$$

$$A2b = \pi\ member\_id, member\_name, member\_belt\ (\sigma\ member\_id = 12345\ MEMBER)$$

$$R2 = \pi\ member\_name, member\_belt, attendance\_date\ (\ A2a \bowtie A2b)$$

Q8. Relational Algebra POST WORKSHOP TASK - answer available Sunday 5PM

The following relations represent a karate dojo member training attendance:

SENSEI (sensei_id, sensei_name)
TRAINING_SCHEDULE (training_day, training_time, group_id, sensei_id)
ATTENDANCE (training_day, training_time, member_id, attendance_date)
MEMBER (member_id, member_name, member_dob, member_belt, group_id)
GROUP (group_id, group_name, group_age_range)

A. Primary keys are underlined
B. A karate member falls into one of the age level groups: Tiny Tiger (for 4-7 year old), Young Dragon (for 8-14 years old), or Adult (for 14+ years old) and owns a certain color of belt (e.g. white, green, brown or black)
C. Sensei (Karate teachers) are scheduled to train an age level group of karate members in a particular day and time (e.g. Sensei Luke Nakamura trains Tiny Tiger members every Tuesday 5pm)
D. A karate member may attend more than one training schedule of their age level group in a given week.

Write the relational algebra for the following query (**your answer must show an understanding of query efficiency**):

(3) Show the id, name and age level group name of members who were absent (did not attend any training) between 01-03-2021 and 31-03-2021 (inclusive).

ANSWER Q8 - POST WORKSHOP TASK

(3) Show the id, name and age level group name of members who were absent (did not attend any training) between 01-03-2021 and 31-03-2021 (inclusive).

Required member details for all members:
A3a = **π** member_id, member_name, group_id (MEMBER)

All member id's who attended:
A3b = **π** member_id (**σ** attendance_date >= 01-03-2021 and
attendance_date <=31-03-2021 (ATTENDANCE))

Required member details for those members who attended:
A3c = A3a ⋈ A3b

All members MINUS members who attended (must be union compatible)
A3d = A3a - A3c

Join with group to get group_id
R = **π** member_id, member_name, group_name (A3d ⋈ (**π** group_id, group_name(GROUP)))

week 4 Normalisation

•理解在数据库设计中进行数据规范化的原因
•理解在数据规范化中各种不同类型的dependency的定义
•理解在数据规范化中从UNF到3NF的各个步骤
•注意到更高阶的数据规范化（Boyce Codd, 4NF和5NF）
•可以通过给定的relation画出dependency图
•可以对给出的数据库描述进行数据规范化的操作
•可以通过给定的逻辑等级的图标进行快速数据规范化

Data Normalisation
▪ Relations MUST be normalised in order to avoid anomalies which may occur when inserting, updating and deleting data.
▪ Normalisation is a systematic series of steps for progressively refining the data model.
▪ A formal approach to analysing relations based on their primary key / candidate keys and functional dependencies

Used:
▪ as a design technique"bottom up design", and
▪ as a way of validating structures produced via "top down design" (ER model converted to a logical model - see next week)
▪ for this unit only concerned with conversion to third normal form

The Normalisation Process Goals

Creating valid relations, i.e. each relation meets the properties of the relational model. In particular:
– Entity integrity – Referential integrity
– No many-to-many relationship
– Each cell contains a single value (is atomic)
创建有效关系，即每个关系都满足关系模型的属性。特别是： – 实体完整性 – 参照完整性 – 没有多对多关系 – 每个单元格包含一个值（是原子的）

In practical terms when implemented in an RDBMS:
– Each table represents a single subject
– No data item will be unnecessarily stored in more than one table (remember some redundancy still exists - minimal redundancy).
– The relationship between tables can be established (via PK and FK pairs).
– Each table is void of insert, update and delete anomalies.
实际上，当在 RDBMS 中实现时： – 每个表代表一个主题 – 没有数据项将不必要地存储在多个表中（记住一些冗余仍然存在 - 最小冗余）。 – 可以建立表之间的关系（通过 PK 和 FK 对）。 – 每个表都没有插入、更新和删除异常。

Representing a form as a relation

This process follows a standard approach:
– arrive at a name for the form which indicates what it represents (its subject)
– determine if any attribute is multivalued (repeating) for a given entity instance of the forms subject if an attribute (or set of attributes) appears multiple times then the group of related attributes need to be shown enclosed in brackets to indicate there are multiple sets of these values for each instance
此过程遵循标准方法： – 为表单指定一个名称，表明其代表什么（其主题） – 确定表单主题的给定实体实例的任何属性是否是多值（重复）如果一个属性（或一组属性）出现多次，则需要将相关属性组显示在括号中，以指示每个实例有多组这些值

example
`DRUG_SLSREP (drug_code, drug_name, slsrep_id, slsrep_name, slsrep_mobile)`
该表格由分配给销售代表数据的重复药品行（实例）组成

Dependency

Functional Dependency Revisited

▪ An attribute B is FUNCTIONALLY DEPENDENT on another attribute A, if a value of A determines a single value of B at any one time.
`CUSTNUMB ➔ CUSTNAME ORDERNO ➔ ORDERDATE`

Functional Dependency

TOTAL DEPENDENCY

– attribute A determines B AND attribute B determines A

`• EMPLOYEE-NUMBER ➔ TAX-FILE-NUMBER • TAX-FILE-NUMBER ➔ EMPLOYEE-NUMBER`

FULL DEPENDENCY

occurs when an attribute is always dependent on all attributes in the composite PK

`ORDERNO, PRODNO ➔ QTYORDERED`

PARTIAL DEPENDENCY

Lack of full dependency for multiple attribute key

`ORDERNO, PRODNO ➔ PRODDESC, QTYORDERED`

– here although qtyordered is fully dependent on orderno and prodno, only prodno is required to determine proddesc

– proddesc is said to be partially dependent on orderno and prodno

TRANSITIVE DEPENDENCY

occurs when Y depends on X, and Z depends on Y - thus Z also depends on X ie. X ➔ Y ➔ Z – and Y is not a candidate key (or part of a candidate key)

`ORDERNO ➔ CUSTNUMB ➔ CUSTNAME`

当 Y 依赖于 X 并且 Z 依赖于 Y 时发生 - 因此 Z 也依赖于 X 即。 X ➔ Y ➔ Z – Y 不是候选键（或候选键的一部分）

Normal Form

UNF

- The UNF representation of a relation is the representation which you have mapped from your inspection of the form
  - it is a **single** named representation (name is not pluralised)
  - no PK etc have as yet been identified
- **ASSIGNMENT** (proj_num, emp_num, emp_name, job_class, chg_hour, assign_hours)
- **ORDER** (orderno, orderdate, custnumb, custname, custaddress (prodno, proddesc, qtyordered, lineprice))

1NF

A RELATION IS IN FIRST NORMAL FORM (1NF) IF:
• a unique primary key has been identified for each tuple/row.
• it is a valid relation
– Entity integrity (no part of PK is null)
– Single value for each cell ie. no repeating group (multivalued attribute).
• all attributes are functionally dependent on all or part of the primary key

如果： • 已为每个元组/行标识了唯一的主键，则关系为第一范式 (1NF)。 • 这是一个有效的关系 – 实体完整性（PK 的任何部分都不为空） – 每个单元格的单个值，即。没有重复组（多值属性）。 • 所有属性在功能上都依赖于全部或部分主键

UNF to 1NF

– identifying a unique identifier for the repeating group.
– remove any repeating group along with the PK of the main relation.
– The PK of the new relation resulting from the removal of repeating group will normally have a composite PK made up of the PK of the main relation and the unique identifier chosen in 1. above, but this must be checked
标识重复组的唯一标识符。
删除任何重复组以及主要关系的 PK。
删除重复组产生的新关系的 PK 通常会有一个复合 PK，由主关系的 PK 和上面 1. 中选择的唯一标识符组成，

UNF

TRAINING (train_code, train_desc, train_active_mnths, (dt_code, dt_model, dt_manuf), (train_date, trainer_id, trainer_rego, trainer_fname, trainer_lname, trainer_category, (cust_id, cust_fname, cust_lname, ct_exam_date, ct_date_expiry)))

Removal of repeating groups working:
TRAINING (train_code, train_desc, train_active_mnths)
DRONETYPE (dt_code, dt_model, dt_manuf, train_code)

TRAINING_COURSE (train_code, train_date, trainer_id, trainer_rego, trainer_fname, trainer_lname, trainer_category, (cust_id, cust_fname, cust_lname, ct_date_start, ct_date_expiry)) - still in UNF has repeating group

INF

TRAINING (train_code, train_desc, train_active_mnths)

DRONETYPE (dt_code, dt_model, dt_manuf, train_code)

TRAINING_COURSE (train_code, train_date, trainer_id, trainer_rego, trainer_fname, trainer_lname, trainer_category)
CKs: (train_code, train_date) and (train_date, trainer_id)

CUST_TRAINING (train_code, train_date, cust_id, cust_fname, cust_lname, ct_exam_date, ct_date_expiry)

Partial Dependency based on Candidate keys (Must use **GENERAL definition**):
trainer_id -> trainer_rego, trainer_fname, trainer_lname, trainer_category
cust_id -> cust_fname, cust_lname

1NF to 2NF

all non key attributes are functionally dependent on the primary key
– all non key attributes are functionally dependent on any candidate key
所有非键属性在功能上依赖于主键
所有非键属性在功能上依赖于任何候选键

2NF

TRAINING (train_code, train_desc, train_active_mnths)

DRONETYPE (dt_code, dt_model, dt_manuf, train_code)

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

TRAINING_COURSE (train_code, train_date, trainer_id)

CUSTOMER (cust_id, cust_fname, cust_lname)

CUST_TRAINING (train_code, train_date, cust_id, ct_exam_date, ct_date_expiry)

Transitive Dependency
No Transitive Dependency

2NF to 3NF

Move from 2NF to 3NF by removing transitive dependencies
– Remove the attributes with transitive dependency into a new relation.
– The determinant will be an attribute in both the original and new relations (it will become a PK / FK relationship)
– Assign the determinant to be the PK of the new relation

删除传递依赖从 2NF 移动到 3NF——将具有传递依赖的属性删除到新关系中。 – 行列式将是原始关系和新关系中的属性（它将成为 PK / FK 关系） – 将行列式分配为新关系的 PK

3NF

TRAINING (train_code, train_desc, train_active_mnths)
DRONETYPE (dt_code, dt_model, dt_manuf, train_code)
TRAINER (trainer_id, trainer_rego, trainer_fname, trainer_lname, trainer_category)
TRAINING_COURSE (train_code, train_date, trainer_id)
CUSTOMER (cust_id, cust_fname, cust_lname)
CUST_TRAINING (train_code, train_date, cust_id, ct_exam_date, ct_date_expiry)

Full dependencies
train_code -> train_desc, train_active_mnths
dt_code -> dt_model, dt_manuf, train_code
trainer_id -> trainer_rego, trainer_fname, trainer_lname, trainer_category
train_code, train_date -> trainer_id
cust_id -> cust_fname, cust_lname
train_code, train_date, cust_id -> ct_exam_date, ct_expiry_date

summary

UNF to 1NF define PK & remove repeating group.
1NF to 2NF remove partial dependency.
2NF to 3NF remove transitive dependency.
UNF 到 1NF 定义 PK 并删除重复组。
1NF 到 2NF 消除部分依赖。
2NF 到 3NF 消除了传递依赖。

Monash Software EMPLOYEE form

- List all attributes found on the form, maintain consistency with previously used attribute names if exist:
  - emp_no, emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode, phone_type, phone_no, degree_name, degree_institution, degree_year, fmemb_no, fmemb_name, fmemb_dob, skill_name
- Determine if any attribute is multivalued (repeating) for a given entity instance
  - phone_type, phone_no, degree_name, degree_institution, degree_year, fmemb_no, fmemb_name, fmemb_dob, skill_name

Monash Software EMPLOYEE form continued

- Group multivalued attributes that are related and place in brackets

EMPLOYEE (emp_no, emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode, (phone_type, phone_no), (degree_name, degree_institution, degree_year), (fmemb_no, fmemb_name, fmemb_dob), (skill_name))

- This is our beginning UNF, to proceed to 1NF:
  - PK of main relation EMPLOYEE is emp_no
  - Four repeating groups to remove
    - Remove repeating group (multi valued attribute/s) along with PK of main relation (here emp_no)
  - assume a phone number may be shared between employees

Monash Software EMPLOYEE form continued

UNF

EMPLOYEE (emp_no, emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode, (phone_type, phone_no), (degree_name, degree_institution, degree_year), (fmemb_no, fmemb_name, fmemb_dob), (skill_name))

1NF

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)

Partial dependencies:

None present

Note we are making an assumption that a phone number may be shared between employees

Monash Software EMPLOYEE form continued

2NF

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)

Transitive dependencies:

None present

Monash Software EMPLOYEE form continued

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)

Full dependencies:

emp_no -> emp_fname, emp_lname, emp_dob, emp_street_no, emp_street, emp_town, emp_pcode

emp_no, phone_no -> phone_type

emp_no, degree_name, degree_institution -> degree_year

emp_no, fmemb_no -> fmemb_name, fmemb_dob

exercise

week 5 Logical Modelling

Define the steps taken to map an ER diagram to a relational model
Map an ER diagram to a relational model
Use SQL Developer - Data Modeler to draw a logical level ER diagram

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

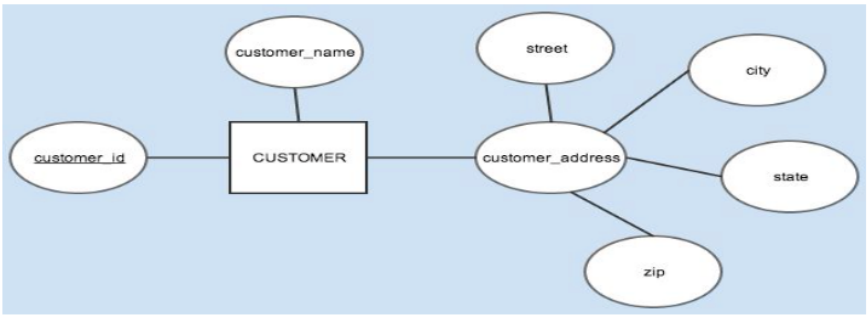
Transforming ER diagrams into relations

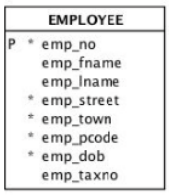
Essentially
– KEY to PK
– Represent relationships with PK/FK pairs

▪ The steps are:
– Map strong (regular) entities
– Map weak entities
– Map binary relationships
– Map associative entities

– Map unary relationships
– Map ternary relationships

Mapping a Composite Attribute

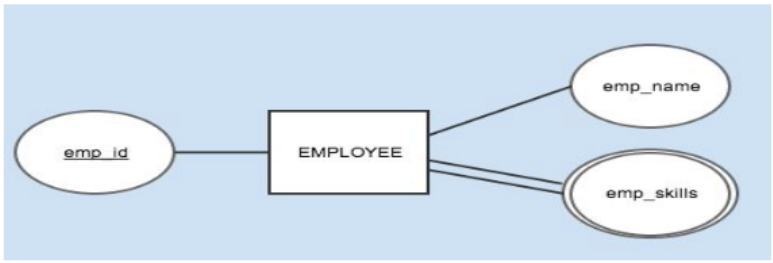




Monash Software Case Study

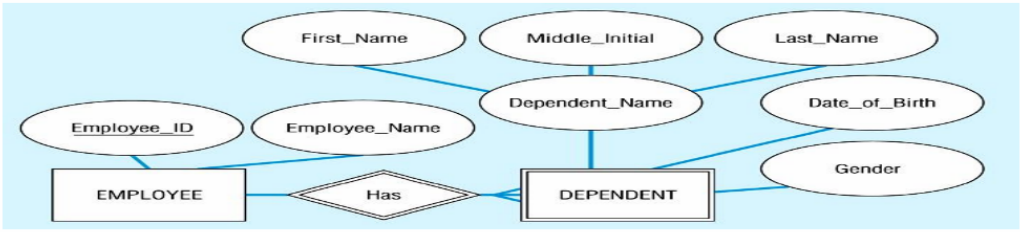
** = not null (must have value)*

Mapping a Multi valued Attribute



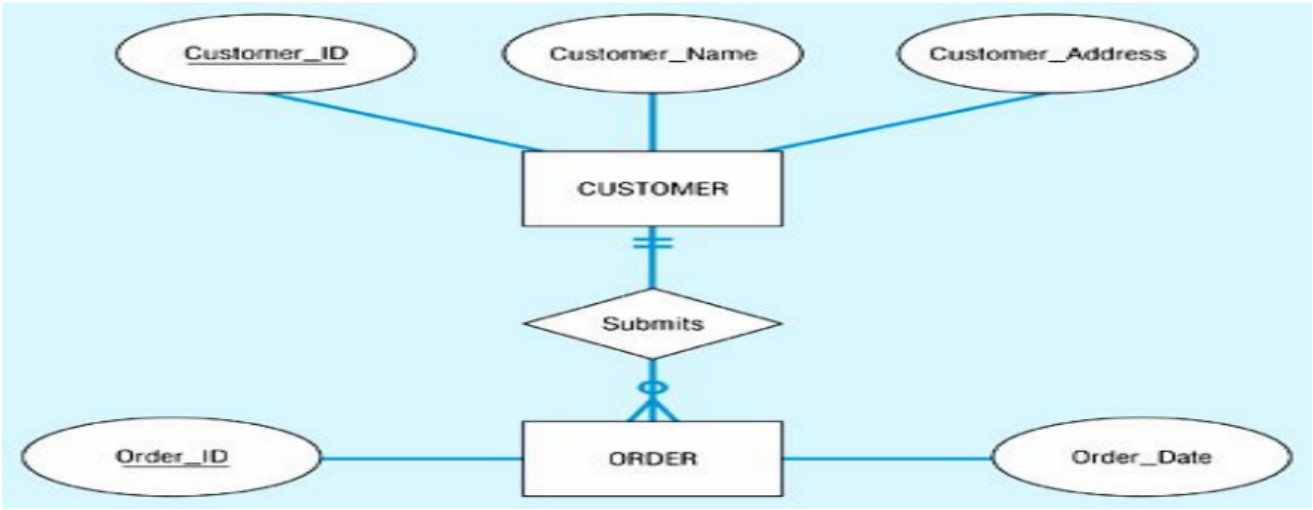
Mapping a Weak Entity

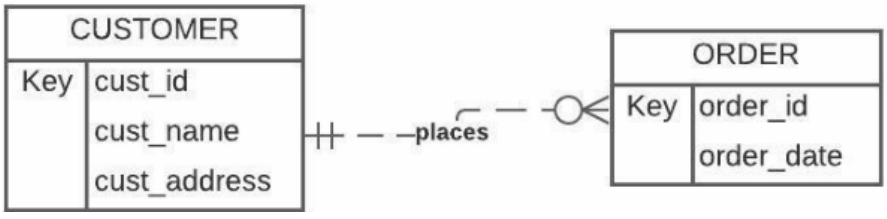
For each weak entity type, create a new relation and include all of the simple attributes as attributes of this relation. The PK of the identifying relation is also included as the FK in this new relation



Mapping a 1:M Binary Relationship

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
对于每个 1:M 二元关系，首先为参与该关系的两个实体类型中的每一个创建一个关系。然后将实体的PK属性（或属性）包含在关系的一侧作为关系的多侧的FK

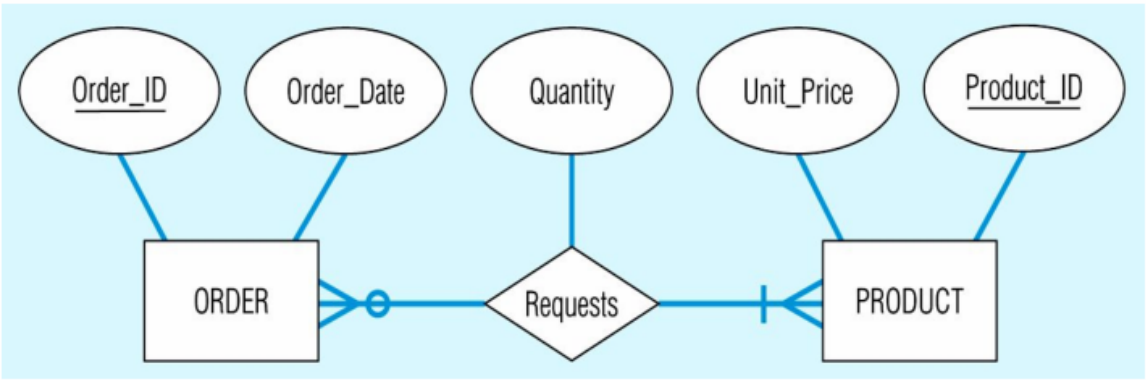


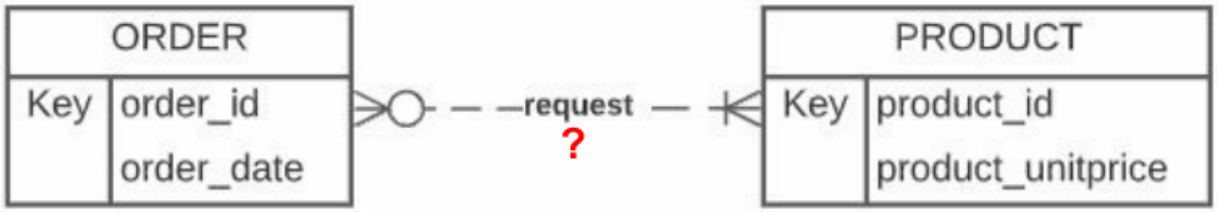


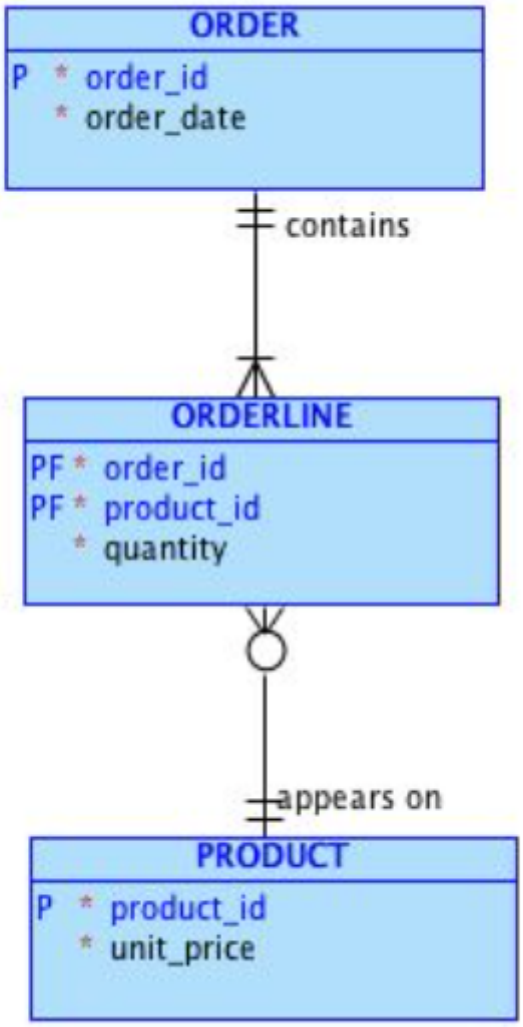
Mapping a M:N Binary Relationship

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

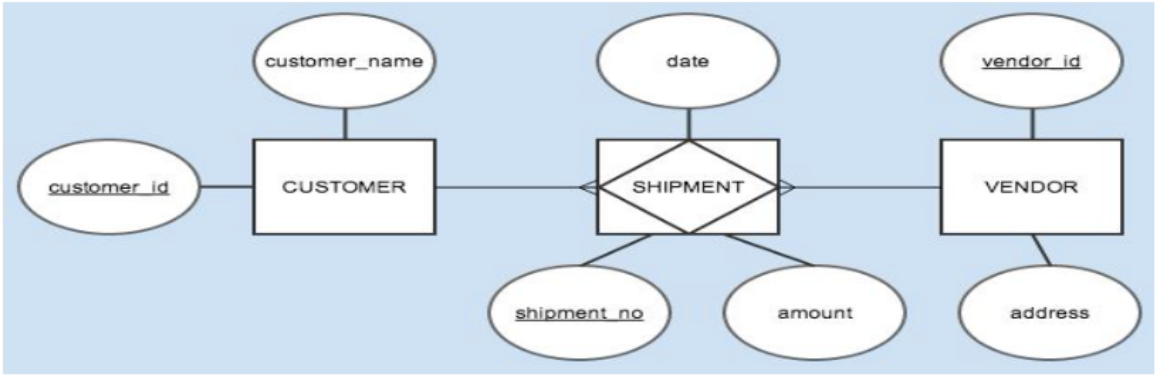
included in the new relation.
首先为参与关系的两个实体类型中的每一个创建一个关系。 – 然后创建一个新关系，并将两个参与实体类型中的每一个的 PK 属性（或多个属性）作为外键属性包括在内。这些属性成为新关系的PK。 – 如果有任何与 M:N 关系关联的非关键属性，它们也包含在新关系中。



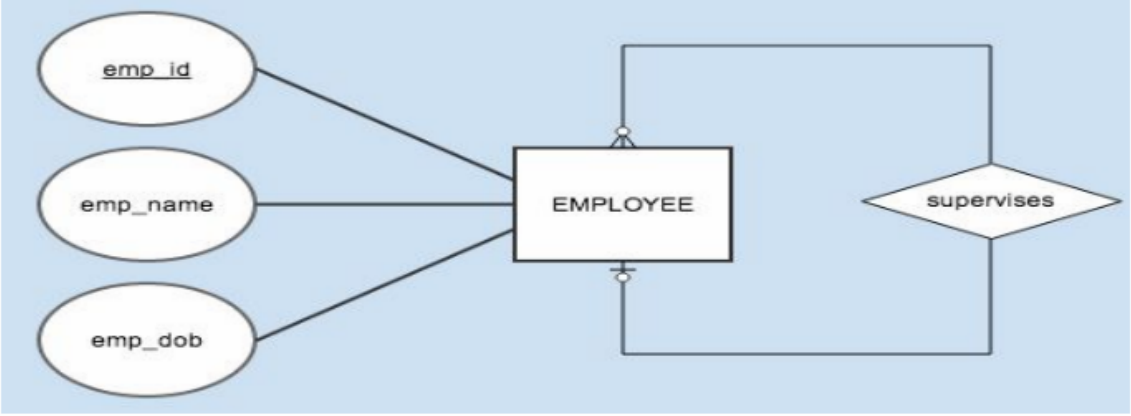


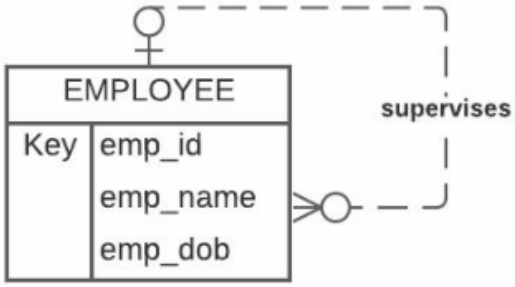


Mapping an associative entity with an Identifier

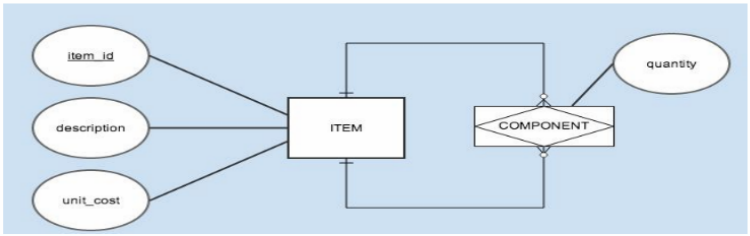


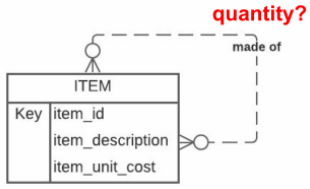
Mapping a 1:M Unary Relationship

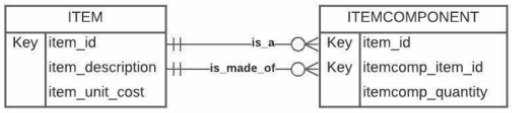


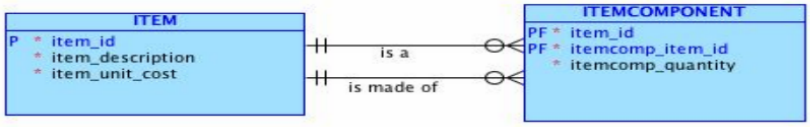


Mapping a M:N Unary Relationship









exercise

week6 Creating & Populating the Database

Map an ER diagram to a relational model

Use SQL Developer - Data Modeler to draw a logical level ER diagram, map it to the relational model and generate a schema file

CREATE A TABLE (DDL)

Column VS Table Level Constraints

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

```
CREATE TABLE training (
   train_code            CHAR(5) NOT NULL,
   train_desc            VARCHAR2(100) NOT NULL,
   train_hrs             NUMBER(2) NOT NULL,
   train_active_months   NUMBER(2) NOT NULL,
   CONSTRAINT training_pk PRIMARY KEY (train_code)
);
```

column constraint

table constraint

Alternative (BETTER) method of defining FKs

```
CREATE TABLE training_course (
   train_code          CHAR(5) NOT NULL,
   traincourse_date        DATE NOT NULL,
   trainer_id              NUMBER(3) NOT NULL,
   CONSTRAINT training_course_pk PRIMARY KEY ( train_code, traincourse_date )
);

ALTER TABLE training_course
   ADD
        ( CONSTRAINT trainer_trainingcourse FOREIGN KEY ( trainer_id )
                REFERENCES trainer ( trainer_id ),
         CONSTRAINT training_trainingcourse FOREIGN KEY ( train_code )
                REFERENCES training ( train_code ));
```

ALTER TABLE

Used to change a tables structure.

- For example:

– Adding column(s).

– Removing column(s).

– Adding constraint(s) - used previously for FK's, but can be any constraint

– Removing constraint(s)

```
ALTER TABLE TRAINER
ADD (CONSTRAINT chk_trainercategory CHECK
            (trainer_category IN ( 'C', 'F' )),
     trainer_nocourses number(3) DEFAULT 0 NOT NULL);
```

ALTER TABLE_Manipulate Constraints

Turn constraint ON or OFF to temporarily disable

```
ALTER TABLE training_course
    DISABLE CONSTRAINT training_trainingcourse;
ALTER TABLE training_course
    ENABLE CONSTRAINT training_trainingcourse;
```

Remove/re add constraint to modify constraint

```
ALTER TABLE training_course
    DROP CONSTRAINT training_trainingcourse;
ALTER TABLE training_course
    ADD
    ( CONSTRAINT training_trainingcourse FOREIGN KEY ( train_code )
    REFERENCES training ( train_code ) ON DELETE CASCADE);
```

DROP_DELETING A TABLE

```
DROP TABLE training_course PURGE
DROP TABLE trainer CASCADE CONSTRAINTS PURGE;
```

ADDING TUPLES/ROWS TO A TABLE (DML)

INSERT_Adding data to table

- SYNTAX:

  INSERT INTO *table [(column [, column...])]*

  VALUES *(value [, value...]);*

```
INSERT INTO training VALUES ('COOO1','Starter Drone Training 1',8,24);

INSERT INTO trainer (trainer_id, trainer_rego, trainer_fname, trainer_lname,
trainer_category) VALUES (312,'DR523412-314','Thomas','Price','F');

INSERT INTO training_course VALUES ('COOO1','20-Oct-2020',312);
```

(TO_DATE)Inserting DATES into a table

convert a date

`to_date('06 Apr 2022','dd Mon yyyy')`

convert a date and time

`to_date('06/04/2022 17:00','dd/mm/yyyy hh:mi')`

convert a time

`to_date('17:00','hh:mi')`

Insert a rental into the RENTAL table

```
insert into RENTAL (rent_no, rent_bond, rent_out, rent_in,drone_id, ct_id)
values (123, 250,to_date('06 Apr 2022 10:00','dd Mon yyyy hh:mi'), null,234,
2345);
```

COMMIT and ROLLBACK

COMMIT makes the changes to the database permanent. ROLLBACK will undo the changes. COMMIT/ROLLBACK only applicable to INSERT/UPDATE and DELETE

(SEQUENCE) auto-increment of a numeric

Create sequence

```
CREATE SEQUENCE manuf_seq
INCREMENT BY 1;
```

Access the sequence using two built-in variables

```
INSERT INTO manufacturer
VALUES(manuf_seq.nextval,'DJI');
```

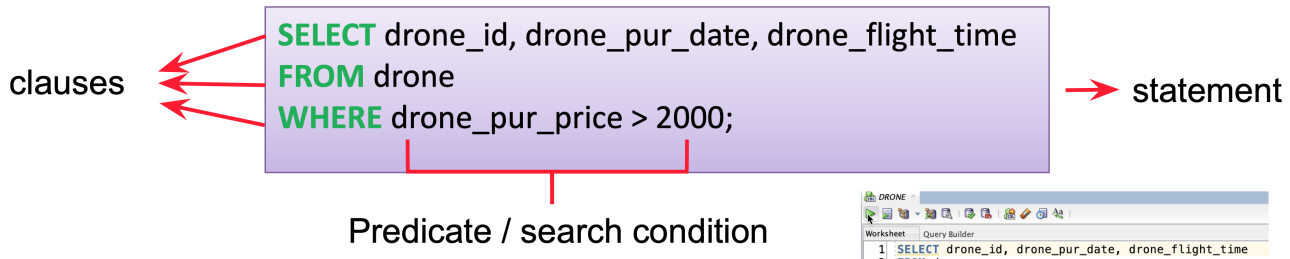
exercise

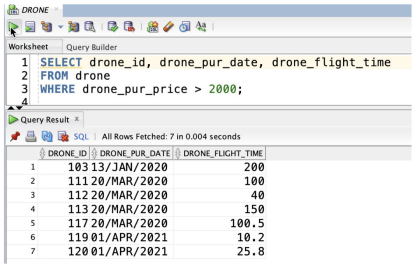
WEEK 7 Structured Query Language (SQL)

Create tables in a database
Add new records to a table
Use ORACLE's sequences to generate keys
Remove tables from a database

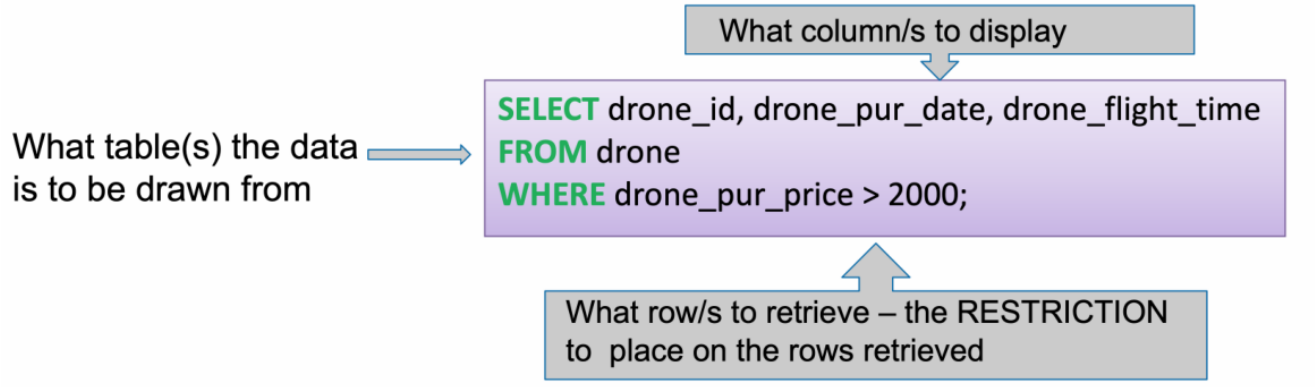
SELECT

Anatomy of an SQL SELECT Statement





Note slides use *tablename* not drone.*tablename* - command as run by drone account user



Comparison

Compare the value of one expression to the value of another expression

`=, !=,< >, <, >, <=, >=`

`drone_pur_price > 2000`

BETWEEN_Range

test whether the value of an expression falls within a specified range of values

`drone_pur_price BETWEEN 3000 AND 5300`

(both are inclusive)

IN_Set Membership

test whether the value of expression equals one of a set of value

`dt_code in ('DMA2','DSPA')`

LIKE_Pattern Match

whether a string (text) matches a specified pattern

% character represents any sequence of zero or more character

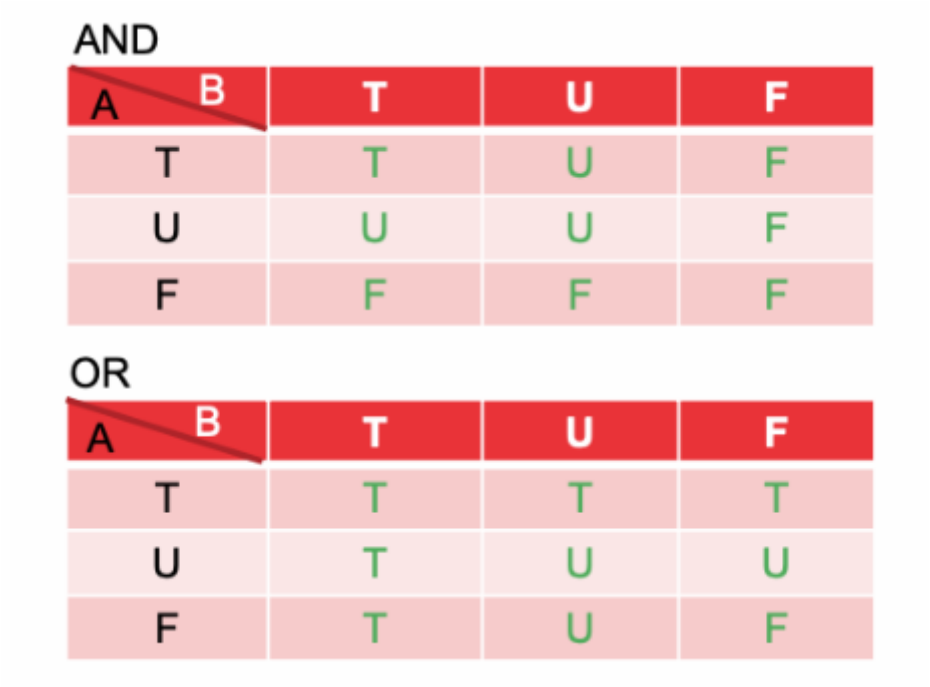
`WHERE dt_model LIKE 'DJI%'`(drone type models starting with DJI)

_ character represents any single character
`WHERE train_code LIKE '__I__'` (drone types with a train_code with an I in the middle)

AND, OR, NOT

An expression is evaluated LEFT to RIGHT
Sub-expression in brackets are evaluated first
NOTs are evaluated before AND and OR
ANDs are evaluated before OR
Use of BRACKETS better alternative

AND is evaluated to be TRUE if and only if both conditions are TRUE
• OR is evaluated to be TRUE if and only if at least one of the conditions is TRUE



NVL_replace a NULL with a value

It is used to replace a NULL with a value (numeric OR character/string)

```
SELECT stu_nbr
NVL(enrol_mark,0),
NVL(enrol_grade,'WH')
FROM enrolment;
```

	STU_NBR	NVL(ENROL_MARK,0)	NVL(ENROL_GRADE,'WH')
1	11111111	78	D
2	11111111	0	WH
3	11111111	0	WH
4	11111112	35	N
5	11111112	0	WH
6	11111113	65	C
7	11111113	0	WH
8	11111114	0	WH

```
select rent_no, drone_id,
to_char(rent_out,'dd-Mon-yyyy') as dateout,
nvl(to_char(rent_in,'dd-Mon-yyyy'),'Still out') as datein
from rental;
```

```
select rent_no, drone_id, rent_out,
nvl(rent_in,'Still out') from rental;
```

RENT_NO	DRONE_ID	DATEOUT	DATEIN
1	100	20-Feb-2020	20-Feb-2020
2	101	21-Feb-2020	22-Feb-2020
3	102	22-Feb-2020	23-Feb-2020
4	100	22-Feb-2020	25-Feb-2020
5	101	25-Feb-2020	25-Feb-2020
6	103	28-Feb-2020	28-Mar-2020
7	103	01-Mar-2020	02-Mar-2020
8	103	03-Mar-2020	04-Mar-2020
9	103	06-Mar-2020	10-Mar-2020
10	101	10-Mar-2020	18-Mar-2020
11	111	26-Apr-2020	28-Apr-2020
12	112	26-Apr-2020	27-Apr-2020
13	113	28-Apr-2020	29-Apr-2020
14	117	28-Apr-2020	05-May-2020
15	103	01-May-2020	02-May-2020
16	103	03-May-2020	10-May-2020
17	112	03-May-2020	07-May-2020
18	113	03-May-2020	12-May-2020
19	118	17-May-2020	18-May-2020
20	118	19-May-2020	23-May-2020
21	118	28-May-2020	29-May-2020
22	118	01-Jun-2020	07-Jun-2020
23	119	11-Apr-2021	Still out
24	120	12-Apr-2021	Still out
25	118	13-Apr-2021	Still out

AS_Renaming Column

```
select drone_id, drone_cost_hr/60 as costpermin from drone;
```

(ORDER BY)Sorting Query Result

Must be used if more than one row may be returned

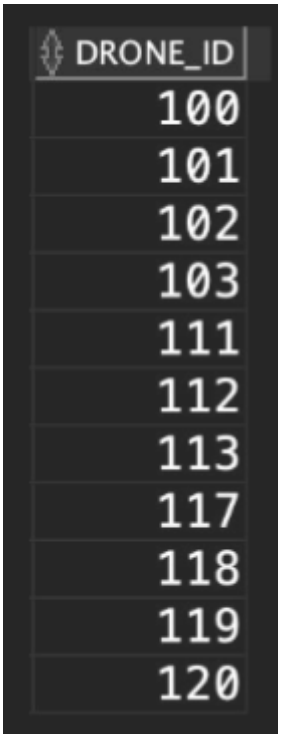
Order can be ASCending or DESCending. The default is ASCending

```
select drone_id, drone_flight_time
from drone
order by
drone_flight_time desc, drone_id;
```

	STU_NBR	ENROL_MARK
1	11111111	(null)
2	11111111	(null)
3	11111114	(null)
4	11111112	(null)
5	11111113	(null)
6	11111111	78
7	11111113	65
8	11111112	35

(DISTINCT)removing Duplicate Rows in the Query Result

```
select distinct drone_id
from rental
order by drone_id;
```



TO_CHAR

Text representing date must be formatted with TO_DATE when comparing or inserting/updating

`select to_char(sysdate,'dd-Mon-yyyy') from dual; = 20-Apr-2021`

`select to_char(sysdate,'dd-Mon-yyyy hh:mi:ss AM') from dual; = 0-Apr-2020 02:51:24 PM`

Worksheet Query Builder

```
SELECT
    drone_id,
    to_char(drone_pur_date, 'dd-Mon-yyyy') AS purchase_date,
    to_char(drone_pur_price, '$99999.99') AS purchase_price,
    to_char(drone_flight_time, '99990.9') AS flight_time
FROM
    drone
WHERE
    drone_pur_date > TO_DATE('01-Mar-2020', 'dd-Mon-yyyy')
ORDER BY
    drone_id;
```

Query Result

SQL | All Rows Fetched: 8 in 0.002 seconds

	DRONE_ID	PURCHASE_DATE	PURCHASE_PRICE	FLIGHT_TIME
1	111	20-Mar-2020	$4200.00	100.0
2	112	20-Mar-2020	$4200.00	40.0
3	113	20-Mar-2020	$4200.00	150.0
4	117	20-Mar-2020	$4200.00	100.5
5	118	01-Apr-2020	$1599.00	56.3
6	119	01-Apr-2021	$5600.80	10.2
7	120	01-Apr-2021	$5600.80	25.8
8	121	17-Apr-2021	$1610.00	0.0

JOIN

placing the join in the where clause is not acceptable

JOIN ON

the general form which always works

`from aaa JOIN bbb ON aaa.a = bbb.b`

`from aaa JOIN bbb ON aaa.key = bbb.key`

TRAINER

TRAINER_ID	TRAINER_REGO	TRAINER_FNAME	TRAINER_LNAME	TRAINER_CATEGORY
1	DR778589-191	Clementius	Cambell	F
2	DR055102-311	Kerwinn	Booeln	C
3	DR322351-719	Charmain	Jado	F
4	DR655884-106	Gaylord	Colegate	F
5	DR820983-603	Garv	Gretton	C

TRAINING_COURSE

TRAIN_CODE	TRAINCOURSE_DATE	TRAINER_ID
DJIHY	14/FEB/2020	1
DJIPR	18/FEB/2020	2
PARPO	25/APR/2020	3
SWELL	10/MAY/2020	4
DJIPR	10/APR/2021	1

Worksheet Query Builder

```
SELECT
    *
FROM
        trainer
    JOIN training_course
    ON trainer.trainer_id = training_course.trainer_id
ORDER BY
    traincourse_date,
    train_code;
```

Query Result

SQL | All Rows Fetched: 5 in 0.004 seconds

	TRAINER_ID	TRAINER_REGO	TRAINER_FNAME	TRAINER_LNAME	TRAINER_CATEGORY	TRAIN_CODE	TRAINCOURSE_DATE	TRAINER_ID_1
1	1	DR778589-191	Clementius	Cambell	F	DJIHY	14/FEB/2020	1
2	2	DR055102-311	Kerwinn	Booeln	C	DJIPR	18/FEB/2020	2
3	3	DR322351-719	Charmain	Jado	F	PARPO	25/APR/2020	3
4	4	DR655884-106	Gaylord	Colegate	F	SWELL	10/MAY/2020	4
5	1	DR778589-191	Clementius	Cambell	F	DJIPR	10/APR/2021	1

JOIN USING

requires matching attribute/s in the two tables

`FROM trainer JOIN training_course USING (trainer_id)`

NATURAL JOIN

requires matching attribute/s in the two tables

`FROM trainer NATURAL JOIN training_course`

TRAINER

TRAINER_ID	TRAINER_REGO	TRAINER_FNAME	TRAINER_LNAME	TRAINER_CATEGORY
1	DR778589-191	Clementius	Cambell	F
2	DR055102-311	Kerwinn	Booeln	C
3	DR322351-719	Charmain	Jado	F
4	DR655884-106	Gaylord	Colegate	F
5	DR820983-603	Garv	Gretton	C

TRAINING_COURSE

TRAIN_CODE	TRAINCOURSE_DATE	TRAINER_ID
DJIHY	14/FEB/2020	1
DJIPR	18/FEB/2020	2
PARPO	25/APR/2020	3
SWELL	10/MAY/2020	4
DJIPR	10/APR/2021	1

Worksheet Query Builder

```
SELECT
    train_code,
    traincourse_date,
    trainer.trainer_id,
    trainer_fname,
    trainer_lname
FROM
        trainer
    JOIN training_course
    ON trainer.trainer_id = training_course.trainer_id
ORDER BY
    traincourse_date,
    train_code;
```

Query Result

SQL | All Rows Fetched: 5 in 0.018 seconds

	TRAIN_CODE	TRAINCOURSE_DATE	TRAINER_ID	TRAINER_FNAME	TRAINER_LNAME
1	DJIHY	14/FEB/2020	1	Clementius	Cambell
2	DJIPR	18/FEB/2020	2	Kerwinn	Booeln
3	PARPO	25/APR/2020	3	Charmain	Jado
4	SWELL	10/MAY/2020	4	Gaylord	Colegate
5	DJIPR	10/APR/2021	1	Clementius	Cambell

Worksheet Query Builder

```
SELECT
    train_code,
    traincourse_date,
    trainer_id,
    trainer_fname,
    trainer_lname
FROM
        trainer
    NATURAL JOIN training_course
ORDER BY
    traincourse_date,
    train_code
```

Query Result

SQL | All Rows Fetched: 5 in 0.003 seconds

	TRAIN_CODE	TRAINCOURSE_DATE	TRAINER_ID	TRAINER_FNAME	TRAINER_LNAME
1	DJIHY	14/FEB/2020	1	Clementius	Cambell
2	DJIPR	18/FEB/2020	2	Kerwinn	Booeln
3	PARPO	25/APR/2020	3	Charmain	Jado
4	SWELL	10/MAY/2020	4	Gaylord	Colegate
5	DJIPR	10/APR/2021	1	Clementius	Cambell

Different Types of SQL JOINs

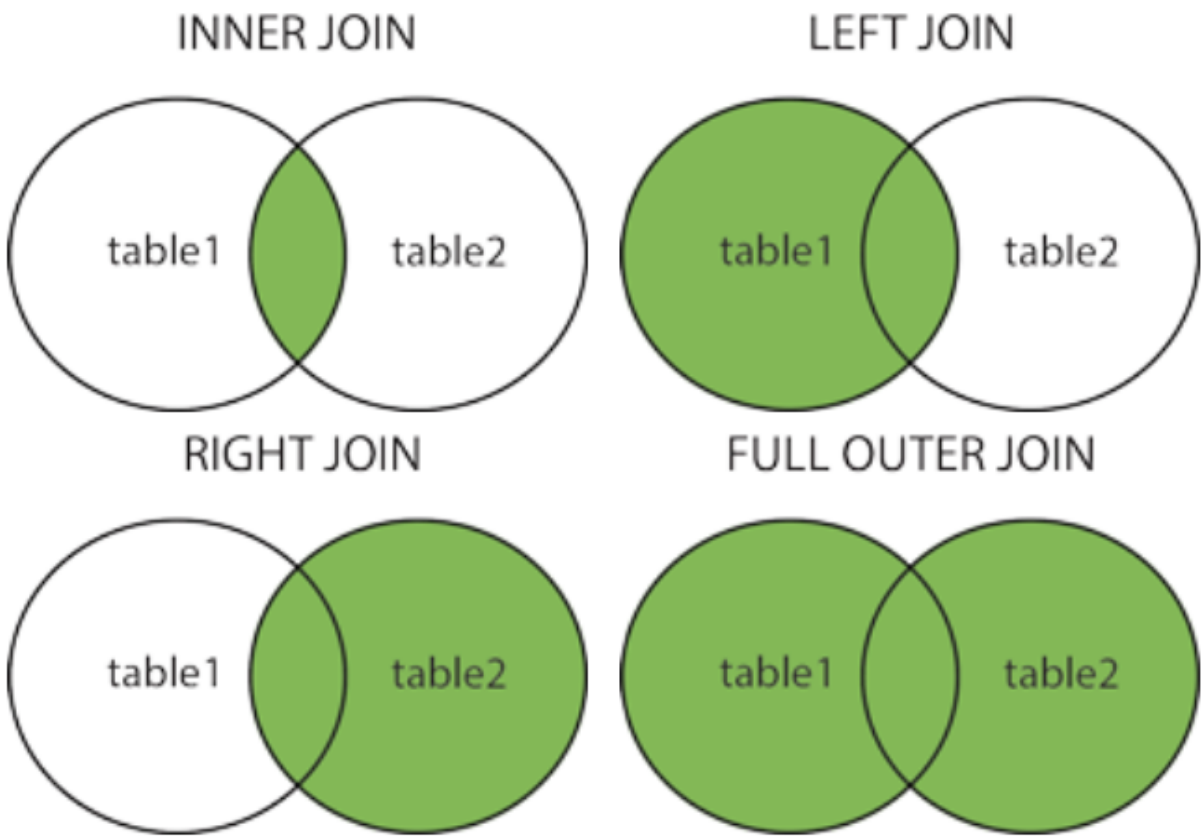
`(INNER) JOIN`: Returns records that have matching values in both tables

`LEFT (OUTER) JOIN`: Returns all records from the left table, and the matched records from the right table

`RIGHT (OUTER) JOIN`: Returns all records from the right table, and the matched records from

the left table

`FULL (OUTER) JOIN`: Returns all records when there is a match in either left or right table



exercise

week8 DML Update, Delete and Transaction Management

interpret a graphical representation of a relational database
● code simple SQL statements on a single table
● code SQL statements that use rows from more than a single table using different types of ANSI standard JOIN operations
● code SQL SELECT statements to select rows based on different conditions
● use ORACLE's date data type in SQL statements correctly
● define an alias for tables and columns
● sort the retrieved data into different orders via SQL ORDER BY
解释关系数据库的图形表示 ● 在单个表上编写简单的 SQL 语句 ● 使用不同类型的 ANSI 标准 JOIN 操作使用多个表中的行编写 SQL 语句 ● 编写 SQL SELECT 语句以根据不同条件选择行● 在 SQL 语句中正确使用 ORACLE 的日期数据类型 ● 为表和列定义别名 ● 通过 SQL ORDER BY 将检索到的数据排序为不同的顺序

FINAL

SQL