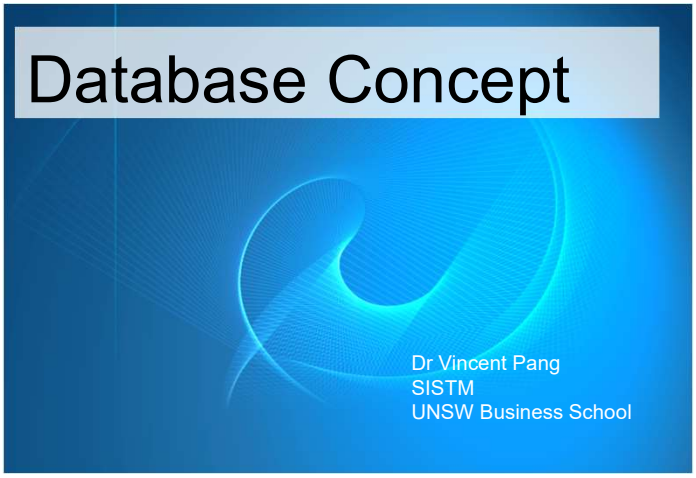
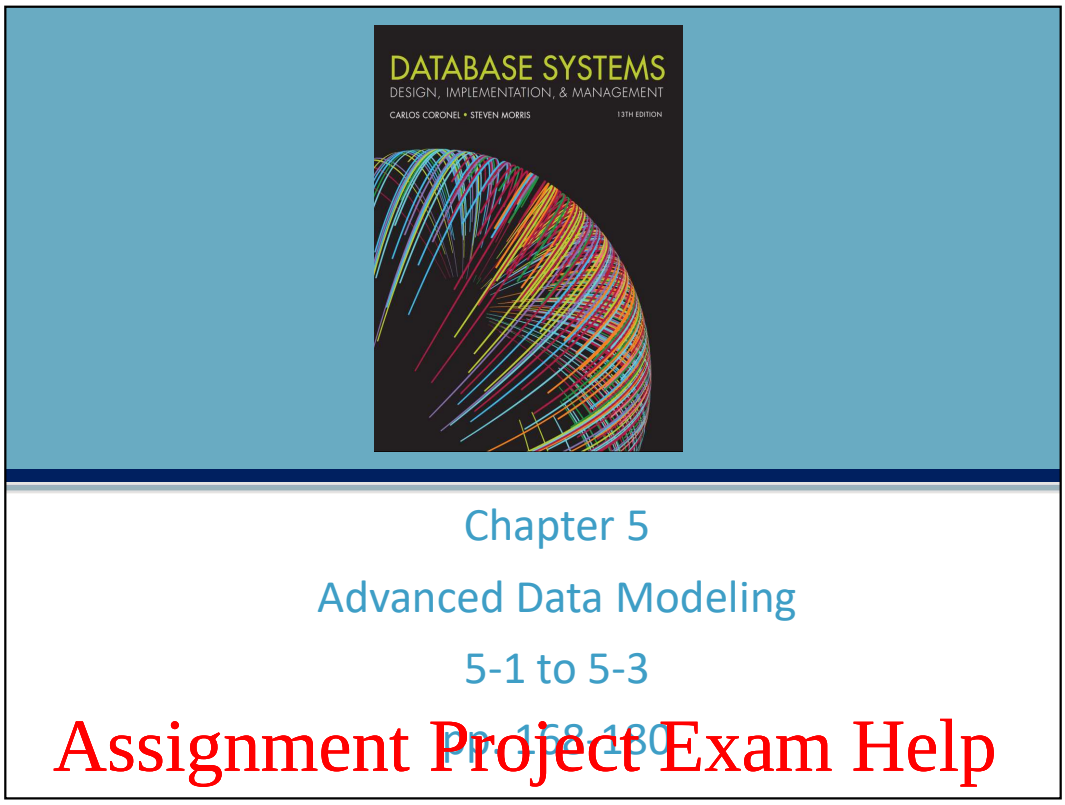
INFS5710 IT Infra. for BA

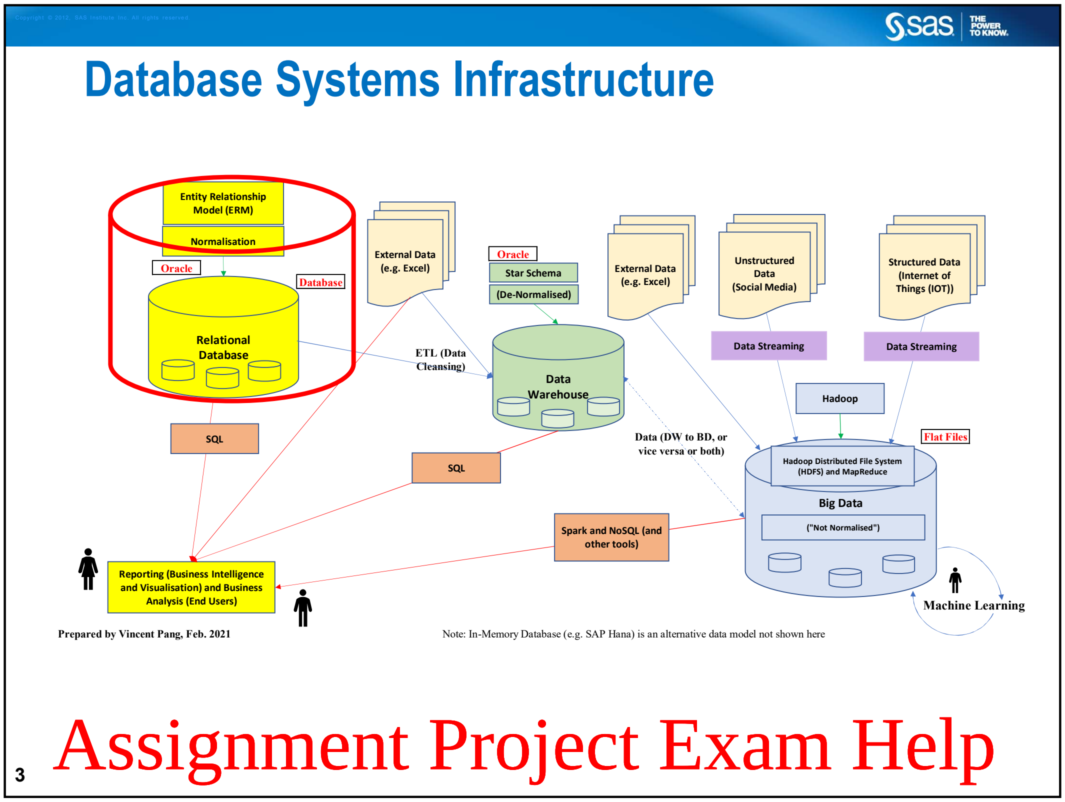




This week we will look at more advanced EER modeling.

Today, some of the designers will totally ignore this section altogether. Why? Because they have in the head already! That is, they know what to do when they implement the physical database.

However, you have to learn because it will help you to think through when designing a database.



The notes in the speaker notes originally are for myself only, but students saw my notes and they asked for them, so I start to share with you all. It is not perfect but hopefully, it makes sense to you!

For the next two weeks, we will be looking at relational database model, particularly Entity Relationship Modelling or ERM. or sometimes we just call it Entity Relationship Diagram or ERD.

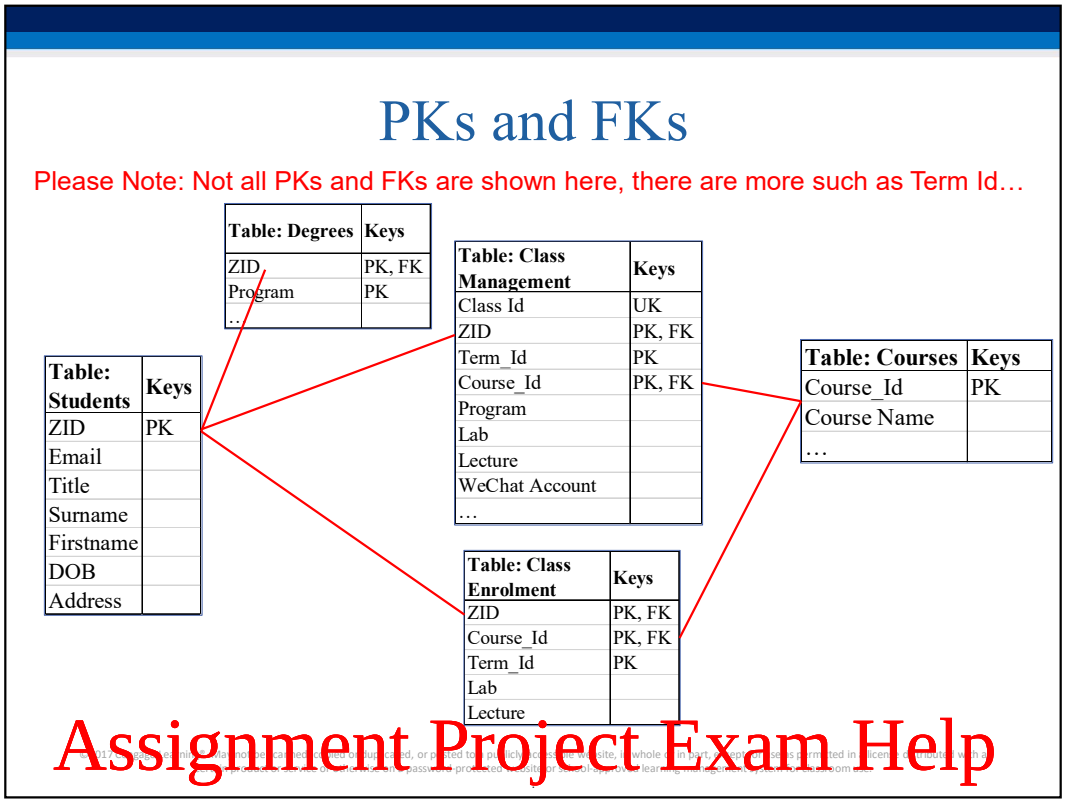
This week we will be looking at relational table, and next week, we will be looking at normalisation of database tables. So, that is the red cylinder.

Extended Entity Relationship Model (EERM)

- Result of adding more semantic constructs to the original entity relationship (ER) model
- **EER diagram (EERD)**: Uses the EER model

4

Extended ER diagram or EERD are looking at more details at the modelling design.



Remember this from previous lecture, you have a student and enrol in several courses in different classes.

Different Input

- What if I were to ask you for your High School results, undergraduate degree, and other degree or diploma, or even your professional membership, I am going to have different results.
- For example, for your high school results, in NSW, you are going to have HSC results and ATAR score. How about in China, India, Indonesia, and UK, they all are different, so how do you keep the records?
- So how about professional memberships? For example, an accountant has a CPA, whereas a plumber need a plumbing certificate (or may be none)

In NSW, you are going to have HSC results and ATAR score.

How about in China, India, Indonesia, and UK, they all are different, so how do you keep the records?

Existence Dependence

Existence dependence: Entity exists in the database only when it is associated with another related entity occurrence

Existence independence: Entity exists apart from all of its related entities, and referred to as a **strong entity** or **regular entity**

Before looking at subtypes, we have a look at the meaning of existence dependence. **As stated in the slide**

A good example of existence dependence is you have parents, and you cannot exist without your parents ☺ We will come back to this.

We will next go through existence independence first.

Existence Independence (1)

Weak (non-identifying) relationship

- Primary key of the related entity does not contain a primary key component of the parent entity

FIGURE 4.8 A WEAK (NON-IDENTIFYING) RELATIONSHIP BETWEEN COURSE AND CLASS

Table name: COURSE — Database name: Ch04_TinyCollege

CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
ACCT-211	ACCT	Accounting I	3
ACCT-212	ACCT	Accounting II	3
CIS-220	CIS	Intro. to Microcomputing	3
CIS-420	CIS	Database Design and Implementation	4
MATH-243	MATH	Mathematics for Managers	3
QM-261	CIS	Intro. to Statistics	3
QM-362	CIS	Statistical Applications	4

Table name: CLASS

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	ROOM_CODE	PROF_NUM
10012	ACCT-211	1	MWF 8:00-8:50 a.m.	BUS311	105
10013	ACCT-211	2	MWF 9:00-9:50 a.m.	BUS200	105
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10015	ACCT-212	1	MWF 10:00-10:50 a.m.	BUS311	301
10016	ACCT-212	2	Th 6:00-8:40 p.m.	BUS252	301
10017	CIS-220	1	MWF 9:00-9:50 a.m.	KLR209	228
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10019	CIS-220	3	MWF 10:00-10:50 a.m.	KLR209	228
10020	CIS-420	1	W 6:00-8:40 p.m.	KLR209	162
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
10022	QM-261	2	TTh 1:00-2:15 p.m.	KLR200	114
10023	QM-362	1	MWF 11:00-11:50 a.m.	KLR200	162
10024	QM-362	2	TTh 2:30-3:45 p.m.	KLR200	162
10025	MATH-243	1	Th 6:00-8:40 p.m.	DRE155	325

For **Weak (non-identifying) relationship**, the Primary Key of the related entity does not contain a primary key component of the parent entity.

In this example, you can see CRS_Code is a primary key of Course table, and CRS_CODE is a foreign key of Class table.

CRS_CODE is a foreign key, but it is not part of the primary key of Class table.

Existence Independence (2)

Strong (identifying) relationships

- Primary key of the related entity contains a primary key component of the parent entity

FIGURE 4.9 A STRONG (IDENTIFYING) RELATIONSHIP BETWEEN COURSE AND CLASS

Table name: COURSE Database name: Ch04_TinyCollege_Alt

CRS_CODE	DEPT_CODE	CRS_DESCRIPTION	CRS_CREDIT
ACCT-211	ACCT	Accounting I	3
ACCT-212	ACCT	Accounting II	3
CIS-220	CIS	Intro. to Microcomputing	3
CIS-420	CIS	Database Design and Implementation	4
MATH-243	MATH	Mathematics for Managers	3
QM-261	CIS	Intro. to Statistics	3
QM-362	CIS	Statistical Applications	4

Table name: CLASS

CRS_CODE	CLASS_SECTION	CLASS_TIME	ROOM_CODE	PROF_NUM
ACCT-211	1	MWF 8:00-8:50 a.m.	BUS311	105
ACCT-211	2	MWF 9:00-9:50 a.m.	BUS200	105
ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
ACCT-212	1	MWF 10:00-10:50 a.m.	BUS311	301
ACCT-212	2	Th 6:00-8:40 p.m.	BUS252	301
CIS-220	1	MWF 9:00-9:50 a.m.	KLR209	228
CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
CIS-220	3	MWF 10:00-10:50 a.m.	KLR209	228
CIS-420	1	W 6:00-8:40 p.m.	KLR209	162
MATH-243	1	Th 6:00-8:40 p.m.	DRE155	325
QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
QM-261	2	TTh 1:00-2:15 p.m.	KLR200	114
QM-362	1	MWF 11:00-11:50 a.m.	KLR200	162
QM-362	2	TTh 2:30-3:45 p.m.	KLR200	162

For **Strong (identifying) relationship**, the Primary key of the related entity contains a primary key component of the parent entity

In this example, you can see CRS_Code is a primary key of Course table, and CRS_CODE is a foreign key of Class table. Moreover, CRS_CODE is also part of the primary key of Class table. CRS_CODE is part of the composite primary key for Class table.

Weak Entities

- **Weak entity** is an entity that **relies on the existence of another (strong or independent) entity**. It has a primary key (PK) that is partially or totally derived from the parent entity in the relationship.
- Weak entity meets two conditions:
  - **Existence-dependent:** Cannot exist without entity with which it has a relationship
  - Has primary key that is **partially or totally derived from the parent entity** in the relationship.
- Database designer usually determines whether an entity can be described as weak **based on the business rules.**

Be careful, weak **entity** is different from **weak relationship**!

Weak entity meets two conditions:

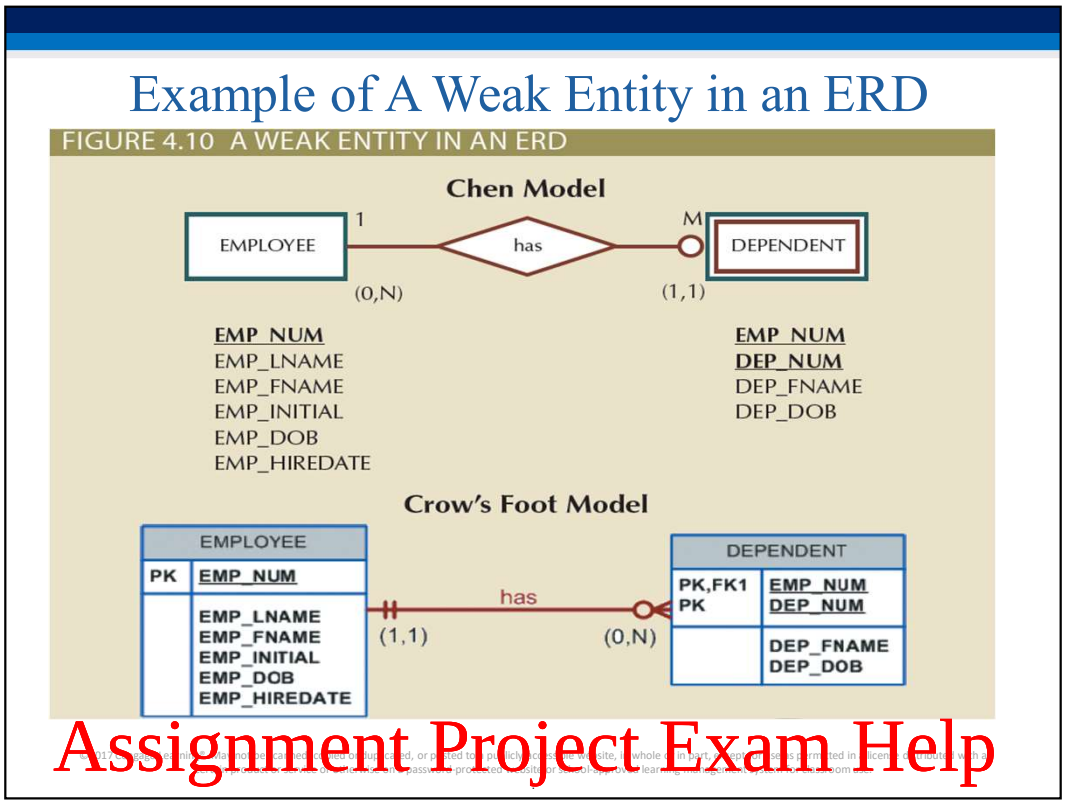
a. **Existence-dependent:** Cannot exist without entity with which it has a relationship

b. Has primary key that is **partially or totally derived from the parent entity** in the relationship.

In UNSW, employee or PG students can put their children into university childcare. As long as the employee or PG student remains at the university, their children can stay at the childcare. However, as soon as the employee or the PG student departs the university, their children have to leave (FYI - that was the old policy, the new policy allows the children to remain in the childcare, but they will be charged at a higher rate).

So that means a weak entity its existence is depending on the existence on another entity in that system. In this case, the other table is Dependent table.

You express that you model that and you are here purely thinking about it from the perspective of the designer. It is based on the business rules.



As stated in the slide

The Dependent entity in Chen model has a composite primary key consists of EMP_NUM and DEP_NUM columns.

An employee has an emp_num, and it is a PK in Employee entity. As for Dependent entity, an employee can have no child, or one child, or two or more children, thus the cardinality is 0:N (0 (zero) because no child). DEP_NUM is a sequential number to keep track number of children. It is optional as an employee does not need to have a child to work in the company. Hence, it is shown as optional relationship.

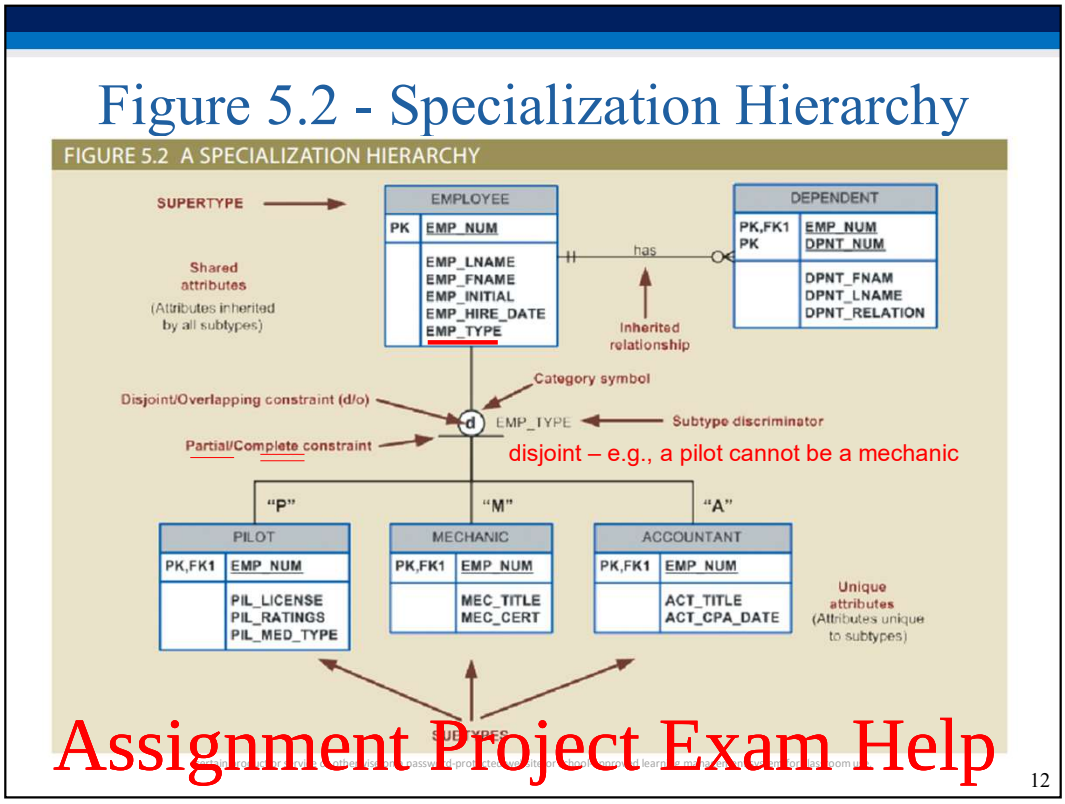
As for fulling the conditions:

a. **Existence-dependent:** Cannot exist without entity with which it has a relationship – in this case it is the Dependent entity. A child must exist with one of his/her parents.

b. Has primary key that is **partially or totally derived from the parent entity** in the relationship – in this case, EMP_NUM in Dependent entity is associated with EMP_NUM of the Employee table.

As for the cardinality relationship for Dependent entity is (1,1) , i.e., a child must have one parent working in the company. If both parents work in the company, you only have to

connect to one. For example, UNSW childcare, it is connected to the parent who will pay childcare fees ☺



Have a look at employee entity, you can see there are three entities join below entity employee.

Employee is a Supertype and it has all the common attributes for all employees, and it has a subtype attribute, or it is called subtype discriminator. In this example, the emp_type has three values, "P", "M", and "A",

In this airline company, there are three entities: one is pilot, one is mechanic, and the last one is accountant. These entities are subtypes.

The employee entity has all the details of employees, but they all work in different jobs. For example, pilot need to have pilot license, but not mechanic and accountant. Employee is the supertype because it has the common attributes of pilot, mechanic, and accountant, which are the subtypes.

Question: what's happen if an accountant has a pilot license as well? Let's see later…

Entity Supertypes and Subtypes

- **Entity supertype**: Generic entity type related to one or more entity subtypes
  - Contains common characteristics
- **Entity subtype**: Contains unique characteristics of each entity subtype
- Criteria to determine the usage
  - There must be different, identifiable kinds of the entity in the user's environment
  - The different kinds of instances should each have one or more attributes that are unique to that kind of instance

EMPLOYEE

PILOT
MECHANIC
ACCOUNTANT

13

As we said in the last slide, Employee entity is the Supertype because it contains common attributes or characteristics for pilot, mechanic, and accountant, which are the subtypes.

Specialization Hierarchy

- Depicts arrangement of higher-level entity supertypes and lower-level entity subtypes
- Relationships are described in terms of "is-a" relationships
- Subtype exists within the context of a supertype
- Every subtype has one supertype to which it is directly related
- Supertype can have many subtypes

An employee "is a" pilot, mechanic, or accountant.

14

Entity supertypes and subtypes are organised into a specialization hierarchy with the high-level entity supertypes, which are the parent entities, and the lower-entity subtypes are the child entities.

The relationships between the entities are described in terms of is-a relationships. So, in our example, an employee is a pilot, mechanic, or accountant.

A subtype must exist within the context of a supertype and must directly associate to one supertype.

A supertype can have many subtypes.

Specialization Hierarchy

- Provides the means to:
  - Support attribute inheritance
  - Define a special supertype attribute known as the subtype discriminator e.g., EMP_TYPE (pilot, mechanic or accountant)
  - Define disjoint/overlapping constraints and complete/partial constraints

Can an employee be a pilot and a mechanic?

Must an employee be a pilot, a mechanic, or an accountant?

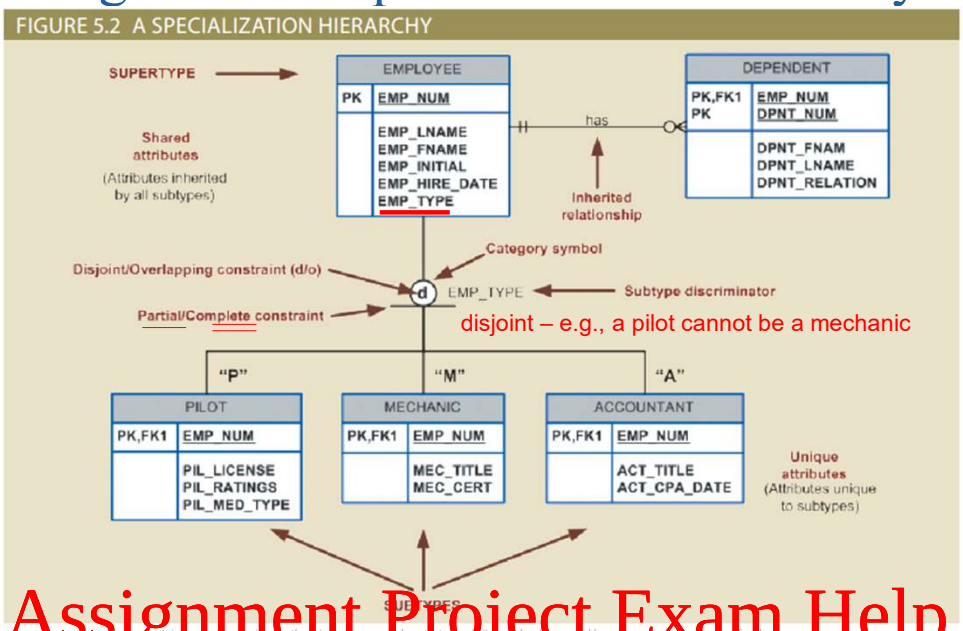
15

It supports attribute inheritance, which means all attributes are inherited by the entity subtypes.

So, how do you differentiate the different subtypes? The way to do is to have a special subtype attribute called subtype discriminator.

In our example, emp_type attribute is the subtype discriminator.

Figure 5.2 - Specialization Hierarchy



If you look if you look here between employee entity and dependant entity, the relationship is "has".

See emp_type attribute in employee entity is the subtype discriminator. So, depending on the value of emp_type attribute, it will branch out differently.

If you look at the category symbol, this is Disjoint, and emp_type is the subtype discriminator. What it means here, for example, is a pilot cannot be a mechanic!

Inheritance

- Enables an entity subtype to inherit attributes and relationships of the supertype
- All entity subtypes inherit their primary key attribute from their supertype
- At the implementation level, supertype and its subtype(s) maintain a 1:1 relationship
- Entity subtypes inherit all relationships in which supertype entity participates
- Lower-level subtypes inherit all attributes and relationships from its upper-level supertypes

17

As stated in the slide

Subtype Discriminator

- Attribute in the supertype entity that determines to which entity subtype the supertype occurrence is related EMP_TYPE is an attribute of the supertype
- Default comparison condition is the equality comparison "is-a" is "="

18

As stated in the slide

Disjoint and Overlapping Constraints

- **Disjoint subtypes**: Contain a unique subset of the supertype entity set

  - Known as **nonoverlapping subtypes**
  - Implementation is based on the value of the subtype discriminator attribute in the supertype
- **Overlapping subtypes**: Contain nonunique subsets of the supertype entity set

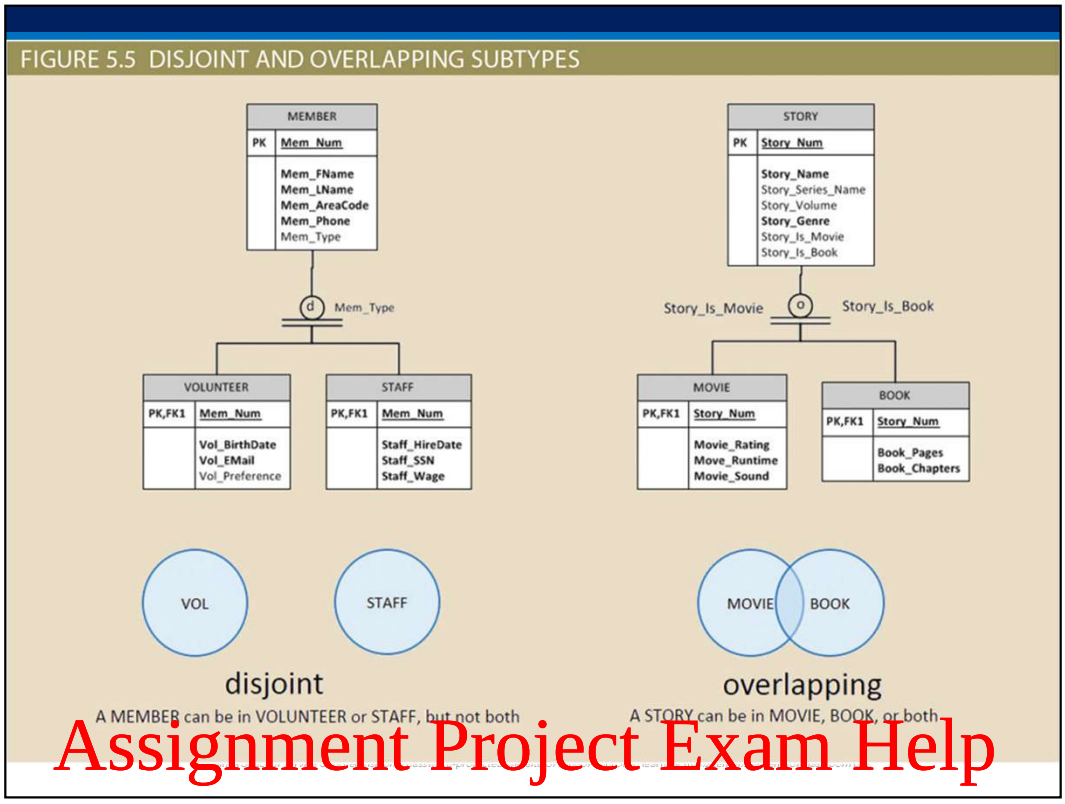
  - Implementation requires the use of one discriminator attribute for each subtype (next slide)

19

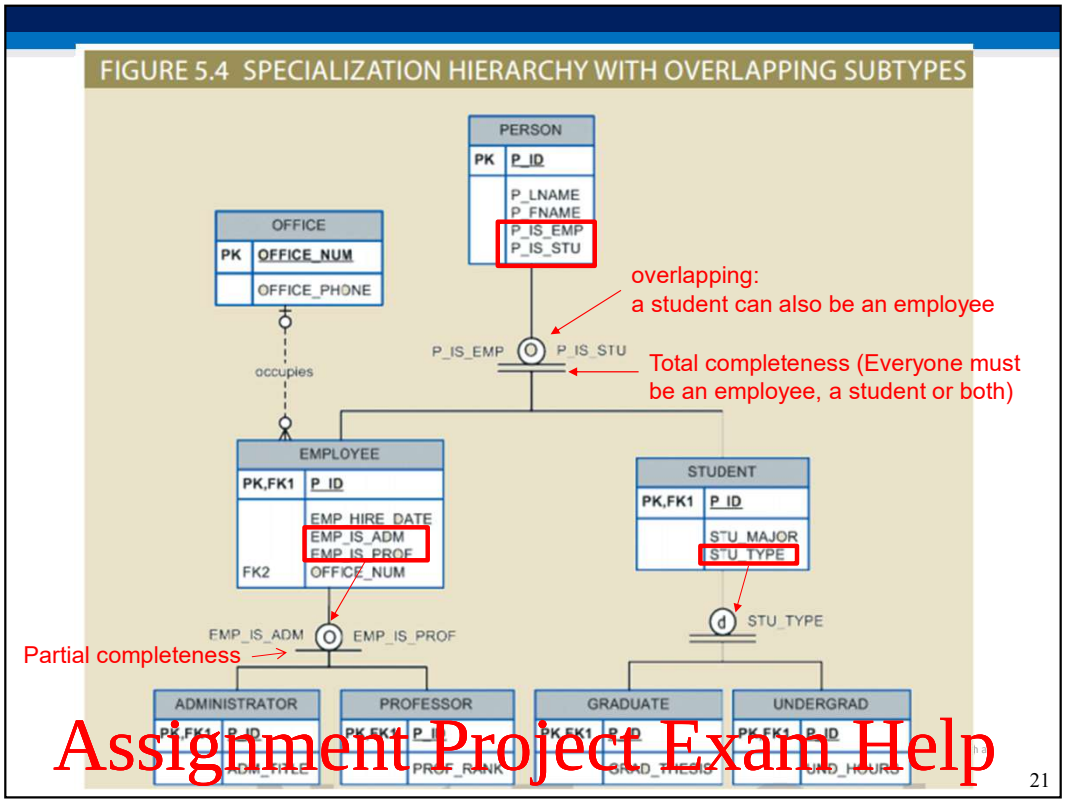
Disjoint subtypes mean the subtypes are independent and not overlapped. For example, P, M and A are not overlapped.

On the other hand, overlapping subtypes are when subtypes entities are overlapped with each other.

See next slide as an example.



For example of Disjoint versus Overlapping? For example, the books of Harry Potters!



If we look at person entity, a student can be an employee, for example you might be working at a university as well as studying.

For Total Completeness, a person can either be an employee, or a student, or both.

That is why you have two attributes, P_IS_EMP and P_IS_STU, one for student and the other for employee. In this University, we used to have a SID for staff and a ZID for student, but with the same number. This is an example off overlapping. But now we only have ZID!

Partial completeness means that not every supertype occurrence is a member of a subtype; some subtype occurrences may not be members of any subtypes

On the other hand, total completeness means that every super subtype occurrence must be a member of at least one subtype. In this example, an employee might not be an administrator or a professor. You might be only a cleaner ☺

Remember the question of an accountant of having a pilot license, depending on the business rules, you can create something like overlapping subtypes. The accountant might be a pilot but spend most of his time doing numbers rather than flying.

Table 5.1 - Discriminator Attributes with Overlapping Subtypes

TABLE 5.1

DISCRIMINATOR ATTRIBUTES WITH OVERLAPPING SUBTYPES

DISCRIMINATOR ATTRIBUTES		COMMENT
PROFESSOR	ADMINISTRATOR	
Y	N	The Employee is a member of the Professor subtype.
N	Y	The Employee is a member of the Administrator subtype.
Y	Y	The Employee is both a Professor and an Administrator.

How about N / N?

22

How about N / N?

This is always partial completeness.

Completeness Constraint

PERSON / EMPLOYEE

- Specifies whether each supertype occurrence must also be a member of at least one subtype
- Types

STUDENT / ADMINISTRATOR
EMPLOYEE / PROFESSOR

  - **Partial completeness**: Not every supertype occurrence is a member of a subtype
  - **Total completeness**: Every supertype occurrence must be a member of any

23

As stated in the slide...

Table 5.2 - Specialization Hierarchy Constraint Scenarios

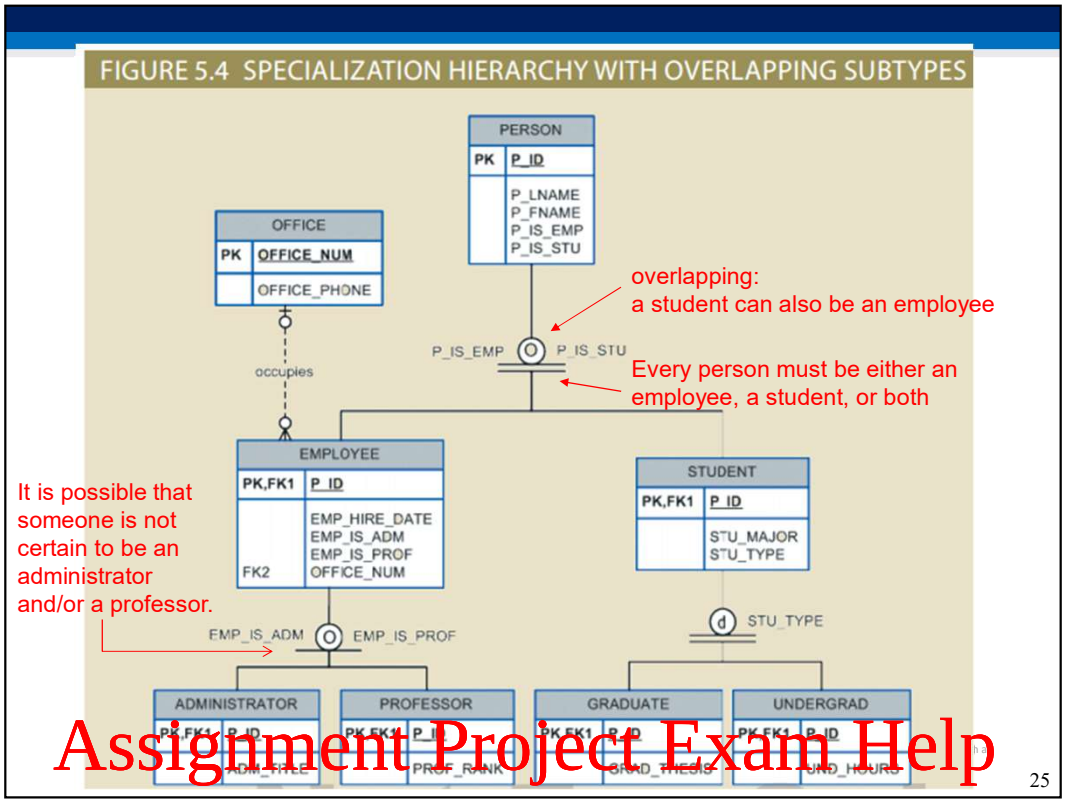
TABLE 5.2

SPECIALIZATION HIERARCHY CONSTRAINT SCENARIOS

TYPE	DISJOINT CONSTRAINT	OVERLAPPING CONSTRAINT
Partial 	Supertype has optional subtypes. Subtype discriminator can be null. Subtype sets are unique.	Supertype has optional subtypes. Subtype discriminators can be null. Subtype sets are not unique.
Total 	Every supertype occurrence is a member of only one subtype. Subtype discriminator cannot be null. Subtype sets are unique.	Every supertype occurrence is a member of atleast one subtype. Subtype discriminators cannot be null. Subtype sets are not unique.

24

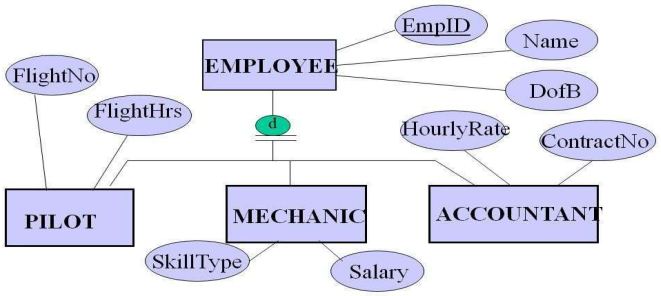
As stated in the slide...



As stated in the slide...

Remember you as an employee might be a cleaner, and not an administrator and/or a professor.

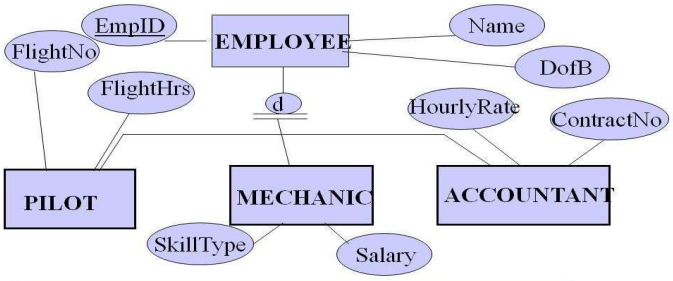
Mapping Super/Subtype Relations (1)



EMPLOYEE (EmpID, Name, DofB, FlightNo, FlightHrs, SkillType, Salary, HourlyRate ContractNo)

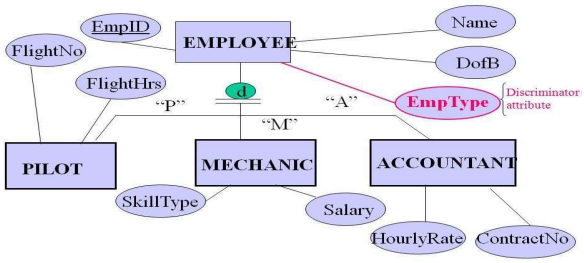
Create only one relation (for the supertype). Or…

Mapping Super/Subtype Relations (2)



PILOT (P_EmpID, Name, DofB, FlightNo, FlightHrs)
MECHANIC (M_EmpID, Name, DofB, SkillType, Salary)
ACCOUNTANT (A_Emp_ID, Name, DofB, HourlyRate, ContractNo)

Mapping Super/Subtype Relations (3)



EMPLOYEE (EmpID, Name, DofB, EmpType)
PILOT (P_EmpID, FlightNo, FlightHrs)
MECHANIC (M_EmpID, SkillType, Salary)
ACCOUNTANT (A_EmpID, HourlyRate, ContractNo)

Create separate relations for each subtype and the supertype.

Primary Keys

Composite key

- Single attribute or a combination of attributes, which uniquely identifies each entity instance
  - Guarantees entity integrity
  - Works with foreign keys to implement relationships

29

As stated in the slide...

Remember, we said last week, your Zid is a primary key.

Natural Keys or Natural Identifier

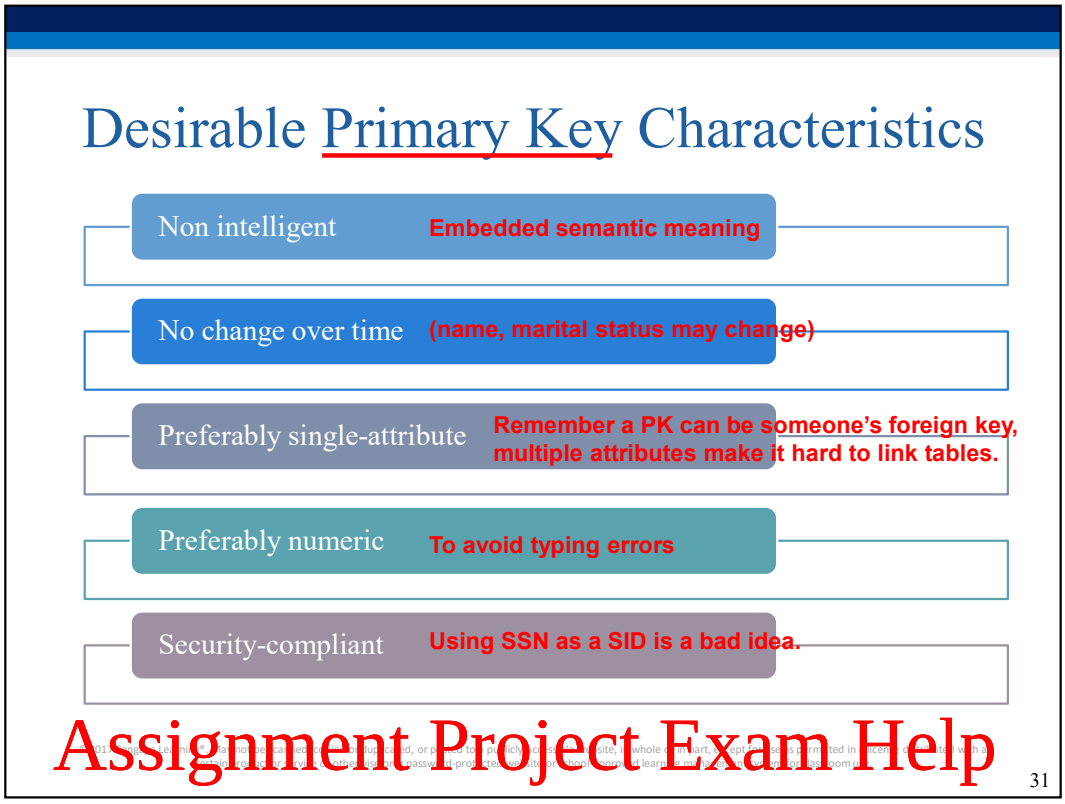
common-sense ones

- Real-world identifier used to uniquely identify real-world objects
  - Familiar to end users and forms part of their day-to-day business vocabulary
  - Also known as natural identifier
  - Used as the primary key of the entity being modeled

Invoice number to identify invoices
Credit card numbers to identify credit cards

30

As stated in the slide...



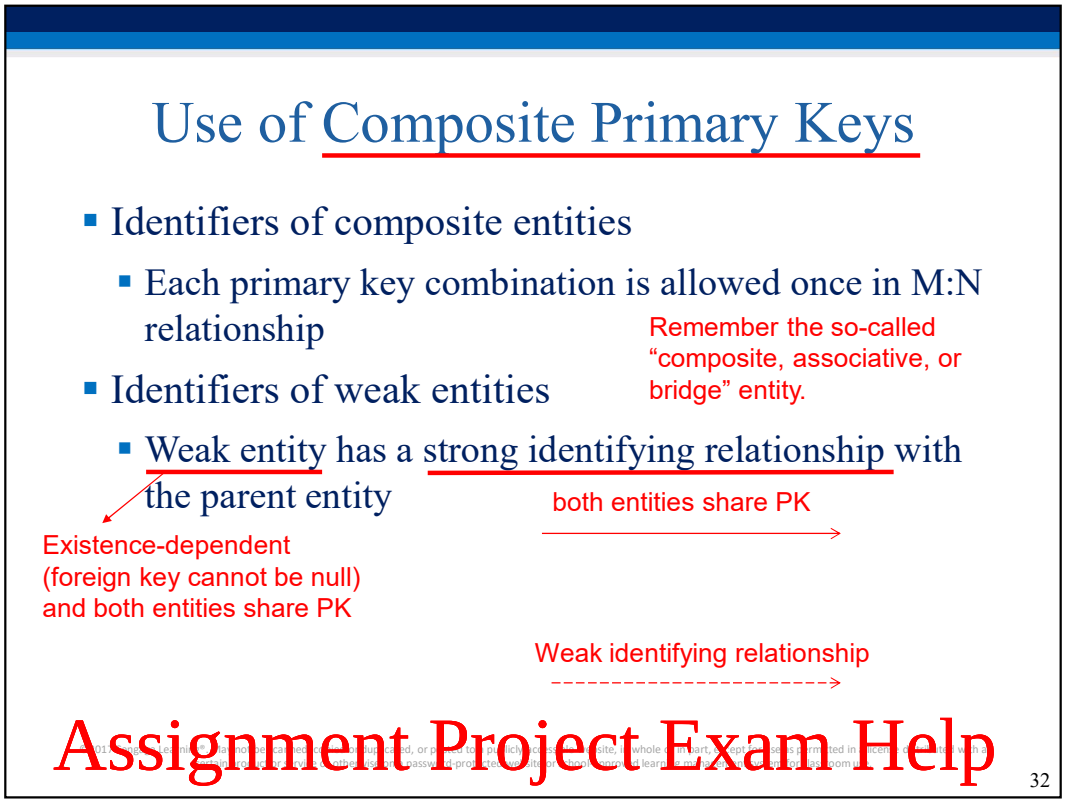
As stated in the slide...

So, how do you select a primary key?

It is best to generate an automatic number with last digit a check digit. A check digit means you have a formula to calculate the check digit based on the digits in the numbers.

For example, the bar code, the last number is a check digit – see https://www.gs1au.org/resources/check-digit-calculator. Once upon a time, I wrote a program to calculate the check-digit, and there are rules of how to calculate the check-digit.

This is better than generating the next number – why?



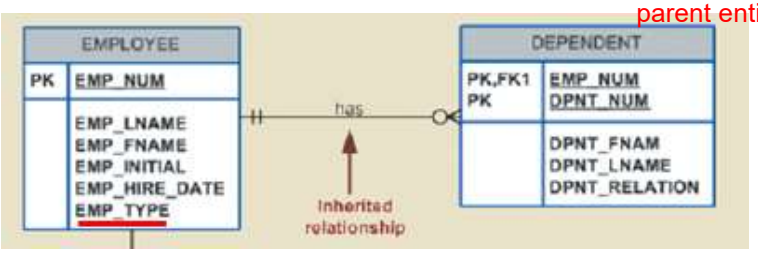
As stated in the slide...

So, when do we create and use composite primary keys? This is when you have many to many relationships, and you want to make it unique, generate composite primary keys is your solution.

Use of Composite Primary Keys

- When used as identifiers of weak entities, represent a real-world object that is: DEPENDENT(emp_num, dpnt_name)
  - Existence-dependent on another real-world object
  - Represented in the data model as two separate entities in a strong identifying relationship

It shares PK from parent entity

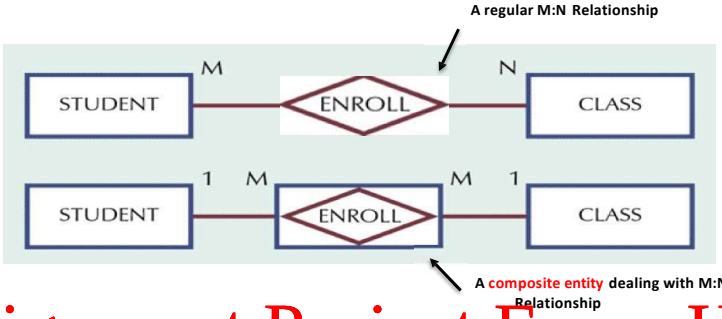


33

For example… as stated in the slide…

Composite Entity

- "A **composite entity** (bridge entity, associative entity) is an entity type that associates the instances of one or more entity types. It contains attributes that are peculiar (singular) to the relationship between those entity instances."
- The composite entity builds a **bridge** between the original entities.
- The composite entity is composed of the **PKs of the original entities**.
- The composite entity may contain **additional attributes.**

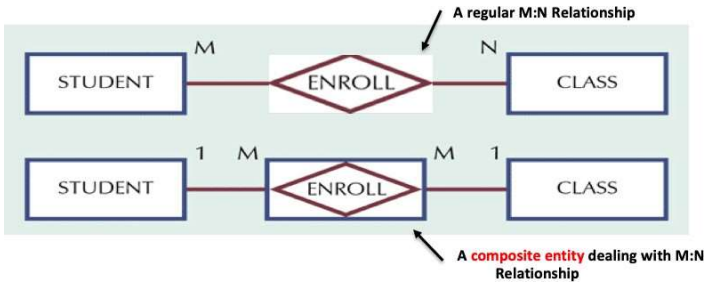


As stated in the slide

The golden rule of dealing with M:N relationship entities – break it down to 1:M relationship using a **composite** entity.

Composite Entity

- **Relational databases** can only handle **1:N relationships** (one-to-many relationships) or **1:1 relationships; M:N relationships** (many-to-many relationships) should be **avoided (via building composite entity)**.
- **A M:N relationship** should be **decomposed** (broken down) to **two** 1:M relationships by creating a **composite entity**.



The relationship between student entity in class entity is **many to many [M:N]**. You cannot create **many to many** relationship tables, so you have to break them down into **one to many** relationship.

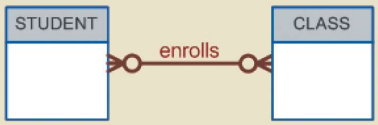
We can change the **Enroll** relationship between **students** entity and **class** entity into a **composite** or **associative** entity. A composite entity deals with many to many relationship. So, we have now got three entities **student** , **class**, and **enroll**.

The relationship between **student** entity and **enroll** entity is one to many. The same for the relationship between **class** entity and **enroll** entity which is also one to many. Thus, having **enroll** entity has resolved the problem of **many to many** relationship, it has broken down into two 1:M relationships. We will look at the attributes later on how this is done.

Many to Many Relationship

In a real world, most things are Many to Many relationships.
You need to resolve into One to Many Relationship.

FIGURE 4.24 THE M:N RELATIONSHIP BETWEEN STUDENT AND CLASS



As stated in the slide

Remember you have M:N relationship for Student and Class/Course!

In a real world, most things are Many to Many relationships. You need to resolve into One to Many Relationship.

Resolve student and Class/Course!

Example of Class Entity in UNSW Timetable

Class Nbr	9819	**Section**	M19A	**Teaching Period**	T1 - Teaching Period One
Activity	Laboratory	**Status**	Open	**Enrols/Capacity**	30/50
Offering Period	17/02/2020 - 17/05/2020	**Meeting Dates**	Standard dates	**Census Date**	15/03/2020
Instruction Mode	In Person	**Consent**	Consent not required		

Meeting Information

Day	Time	Location	Weeks	Instructor
Mon	19:00 - 21:00	Mathews 105 (K-F23-105)	1-8,10-11	Dr VY Pang,Mr C Stead

Class Notes

Class Nbr	9820	**Section**	T10A	**Teaching Period**	T1 - Teaching Period One
Activity	Laboratory	**Status**	Open	**Enrols/Capacity**	22/50
Offering Period	17/02/2020 - 17/05/2020	**Meeting Dates**	Standard dates	**Census Date**	15/03/2020
Instruction Mode	In Person	**Consent**	Consent not required		

Meeting Information

Day	Time	Location	Weeks	Instructor
Tue	10:00 - 12:00	UNSW Business School G26 (K-E12-G26)	1-10	Dr VY Pang,Mr C Stead

This is an example of the Class entity from previous page. This is an example in the UNSW system, and you can see from your UNSW Timetable.

Please note that the Class entity from previous slide is simple, and in real UNSW system, it is slightly more complex, some of the entities are broken down even further. We will talk about this next week in Normalisation.

Class_code: Class Number – e.g. 9819 and 9820

Class_time: A combination of day and time – 7pm to 9pm and 10am to 12pm

Room_code: Not shown but Room Description is shown – Matthews 105 and UNSW BS G26.

Prof_num: Not shown but Instructor Names are shown (here, it breaks down even further – you can have more than one name)

etc…

Note

Each room also has a record keeping on how many students enrolled – this is an example of ***derived attribute***. Normally, you do not save this number, because it can change depending on how many students enrolled in the class. However, I guess this attribute (column) is

created in this table purely for a quick retrieving for display on a report.

Resolve M:N Relationship

FIGURE 5.7 THE M:N RELATIONSHIP BETWEEN STUDENT AND CLASS

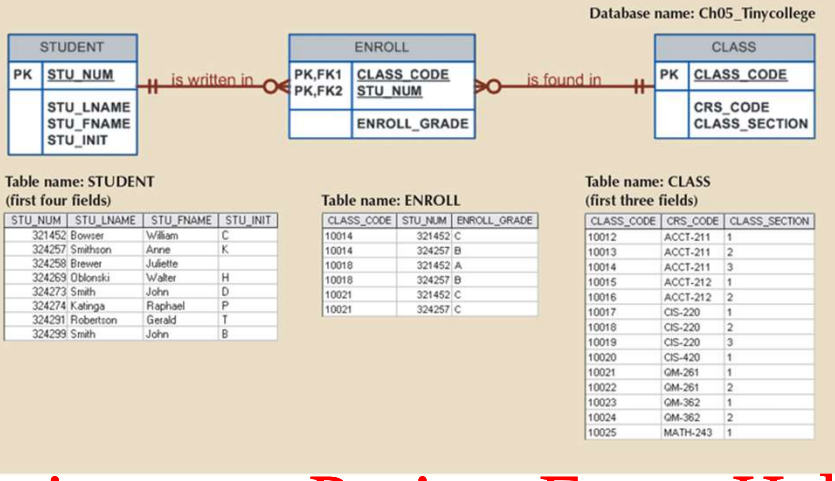


Table name: STUDENT
(first four fields)

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT
321452	Bowser	William	C
324257	Smithson	Anne	K
324258	Brewer	Juliette	
324269	Oblonski	Walter	H
324273	Smith	John	D
324274	Katinga	Raphael	P
324291	Robertson	Gerald	T
324299	Smith	John	B

Table name: ENROLL

CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

Table name: CLASS
(first three fields)

CLASS_CODE	CRS_CODE	CLASS_SECTION
10012	ACCT-211	1
10013	ACCT-211	2
10014	ACCT-211	3
10015	ACCT-212	1
10016	ACCT-212	2
10017	CIS-220	1
10018	CIS-220	2
10019	CIS-220	3
10020	CIS-420	1
10021	QM-261	1
10022	QM-261	2
10023	QM-362	1
10024	QM-362	2
10025	MATH-243	1

Resolve the M:N relationship by creating a new enroll entity containing a composite key

So you need to:

Looking at the attributes:

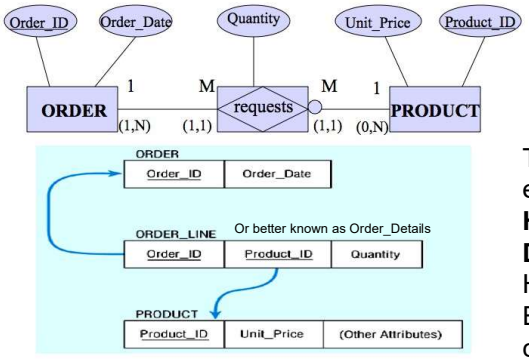
The student entity has STU_NUM is the primary key.

The class entity has class_code as the primary key. (Have a look at next page).

The Enroll entity has a composite key of class_code and STU_NUM.

Another example Breaking down of M:N Relationship

Breaking down M:N Relationship entities into 1:M Relationship entities by introducing composite entity



This is an example of **Header – Details Entities.** Header – Details Entities are very common (more later…)

ER relationship forms a relation in itself

39

Surrogate Primary Keys

- Primary key used to simplify the identification of entity instances are useful when:
  - There is no natural key
  - Selected candidate key has embedded semantic contents or is too long (e.g., composite PK)
- Require ensuring that the candidate key of entity in question performs properly (see an example on the next slide)
  - Use unique index and not null constraints

40

This is best to explain best explain as an example, so see next slide.

Table 5.4 - Data Used to Keep Track of Events

TABLE 5.4

DATA USED TO KEEP TRACK OF EVENTS

DATE	TIME_START	TIME_END	ROOM	EVENT_NAME	PARTY_OF
6/17/2016	11:00a.m.	2:00p.m.	Allure	Burton Wedding	60
6/17/2016	11:00a.m.	2:00p.m.	Bonanza	Adams Office	12
6/17/2016	3:00p.m.	5:30p.m.	Allure	Smith Family	15
6/17/2016	3:30p.m.	5:30p.m.	Bonanza	Adams Office	12
6/18/2016	1:00p.m.	3:00p.m.	Bonanza	Boy Scouts	33
6/18/2016	11:00a.m.	2:00p.m.	Allure	March of Dimes	25
6/18/2016	11:00a.m.	12:30p.m.	Bonanza	Smith Family	12

These are required fields to uniquely identify an event. It is too cumbersome to use them all as a foreign key. Programmer may like to assign an "event_id" for each event, not known to the user, but only to the programmer for convenience, which is a surrogate PK.

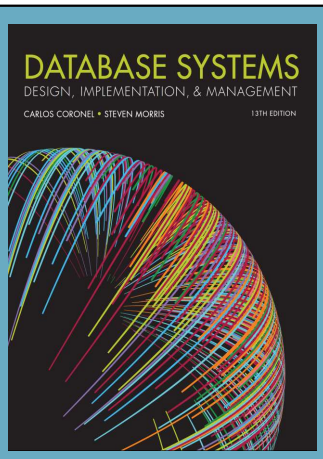
41

If you look at the example, you can see to make this event unique, you have to create a Composite key containing date, time start, and room.

It might be too cumbersome to use them all as a foreign key.

You can create an id called event_id composites these attributes. This is called a surrogate primary key. It is not for programmer convenience but also performance of the database.

However, if you use a surrogate primary key, you still need to check for uniqueness of other attributes. You might have a unique key to ensure these attributes are unique! So you might have a primary key and a unique key – this is more to do with designing a physical database.

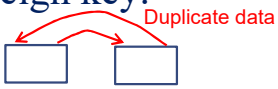


5-4 Design Cases: Learning Flexible Database Design

We will not cover in the lecture, you can read this yourself in the textbook!

Design Case 1: Implementing 1:1 Relationships

- Foreign keys work with primary keys to properly implement relationships in relational model
- Rule
  - Put primary key of the parent entity on the dependent entity as foreign key √
- Options for selecting and placing the foreign key:
  - Place a foreign key in both entities ×
  - Place a foreign key in one of the entities √



43

As stated in the slide and read the textbook for explanations.

TABLE 5.5

SELECTION OF FOREIGN KEY IN A 1:1 RELATIONSHIP

CASE	ER RELATIONSHIP CONSTRAINTS	ACTION
I	One side is mandatory and the other side is optional.	Place the PK of the entity on the mandatory side in the entity on the optional side as a FK, and make the FK mandatory.
II	Both sides are optional.	Select the FK that causes the fewest nulls, or place the FK in the entity in which the (relationship) role is played.
III	Both sides are mandatory.	See Case II, or consider revising your model to ensure that the two entities do not belong together in a single entity.

FIGURE 5.7 THE 1:1 RELATIONSHIP BETWEEN DEPARTMENT AND EMPLOYEE

A One-to-One (1:1) Relationship:
An EMPLOYEE manages zero or one DEPARTMENT;
each DEPARTMENT is managed by one EMPLOYEE.

44

As stated in the slide and read the textbook for explanations.

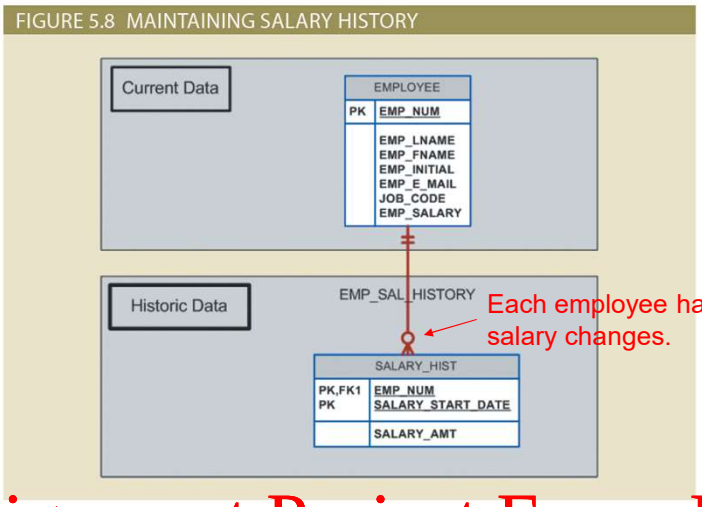
Design Case 2: Maintaining History of Time-Variant Data

- **Time-variant data**: Data whose values change over time and for which a history of the data changes must be retained
  - Requires creating a new entity in a 1:M relationship with the original entity
  - New entity contains the new value, date of the change, and other pertinent attribute

45

As stated in the slide and read the textbook for explanations.

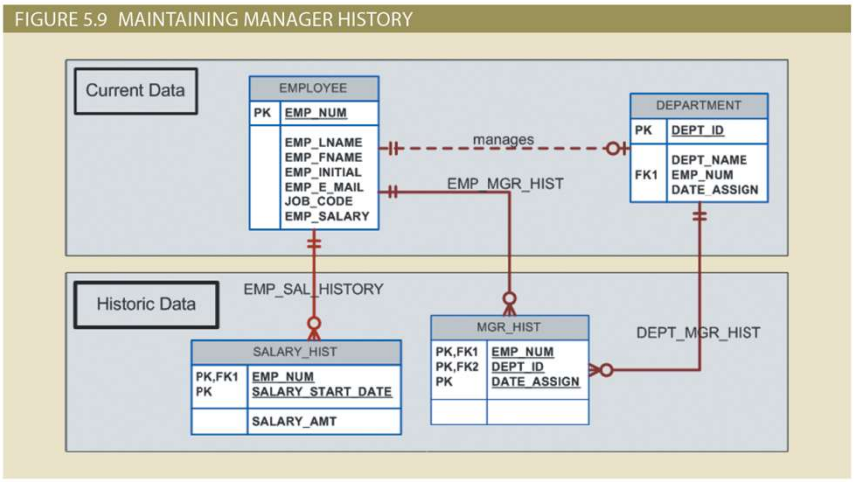
Figure 5.8 - Maintaining Salary History



46

As stated in the slide and read the textbook for explanations.

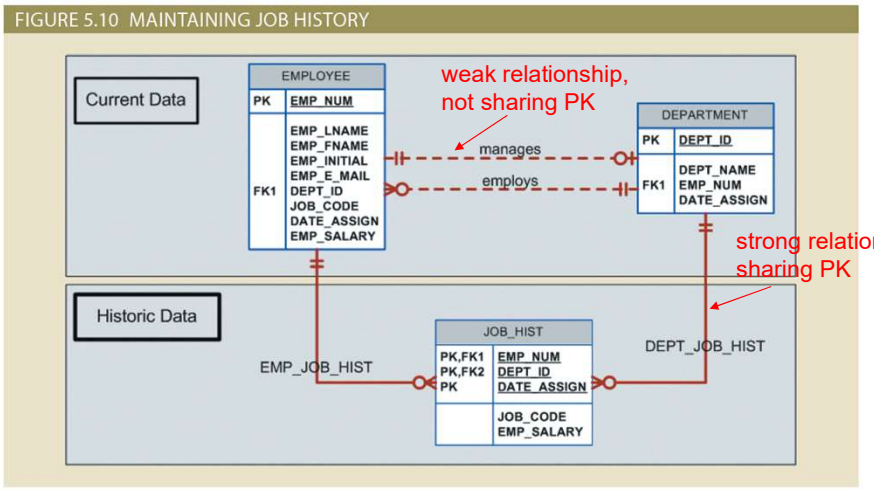
Figure 5.9 - Maintaining Manager History



47

As stated in the slide and read the textbook for explanations.

Figure 5.10 - Maintaining Job History



48

As stated in the slide and read the textbook for explanations.

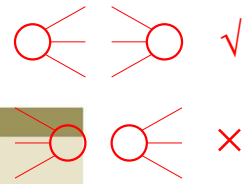
Design Case 3: Fan Traps

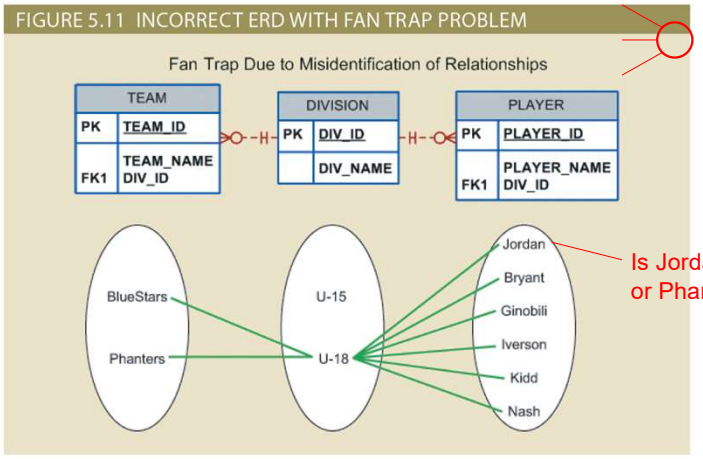
- **Design trap**: Occurs when a relationship is improperly or incompletely identified
  - Represented in a way not consistent with the real world
- **Fan trap**: Occurs when one entity is in two 1:M relationships to other entities
  - Produces an association among other entities not expressed in the model

49

As stated in the slide and read the textbook for explanations.

Figure 5.11 - Incorrect ERD with Fan Trap Problem

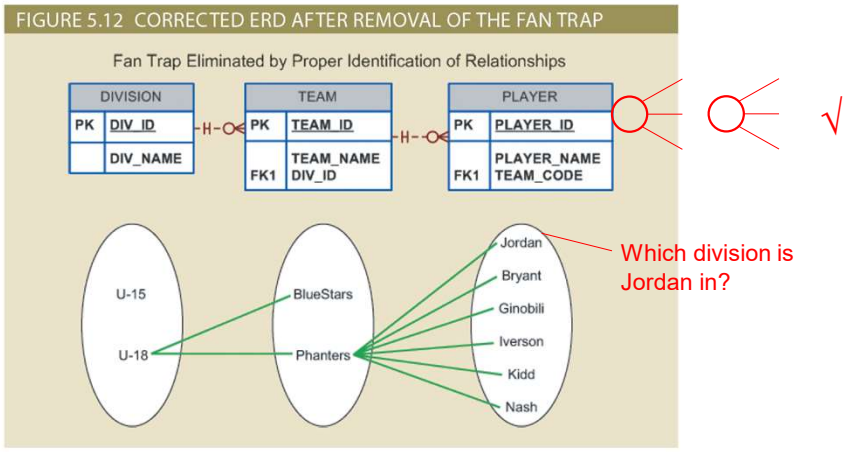




50

As stated in the slide and read the textbook for explanations.

Figure 5.12 - Corrected ERD After Removal of the Fan Trap



51

As stated in the slide and read the textbook for explanations.

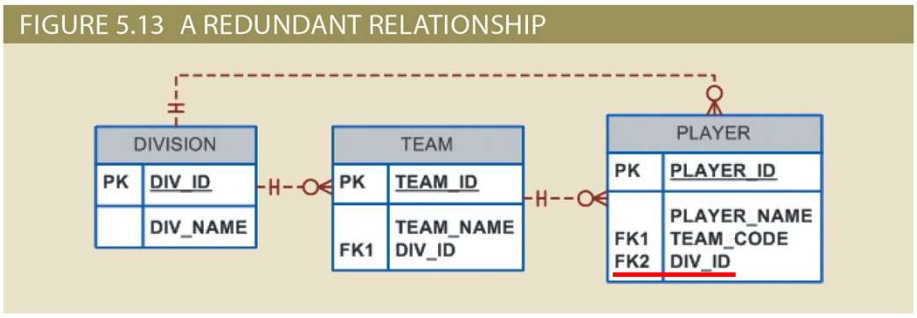
Design Case 4: Redundant Relationships

- Occur when there are multiple relationship paths between related entities
- Need to remain consistent across the model
- Help simplify the design Design trade-off

52

As stated in the slide and read the textbook for explanations.

Figure 5.13 - A Redundant Relationship



53

As stated in the slide and read the textbook for explanations.

This is a common error - if you know which division a player plays in by joining to table entity and division entity.