Data management: be used to capture, store, retrieve, analyze, present and interpret (large amount of) data

**Database**: an organized collection of related data, usually stored on disk

* Important data
* Shared
* Secured
* Well-designed (minimal redundancy)
* Variable size

**Database Management System (DBMS)**: a software system designed to store and manage databases

Usages:

* Maintain integrity (integrity constraints)
* Control redundancy
* Create, modify, and query a database
* Provide support for decision making
* Control access

Different user groups may have different access privileges (create/ alter, update, and retrieve), which are controlled through DBMS security sub-system, through the use of accounts & passwords

* Casual users – may not have access to confidential data
* Parametric users – may be given update access, but are generally not allowed to change the structure of data
* Database administrators (DBAs) – generally have highest privileges, create user accounts and enforce restrictions
* Permit concurrent/simultaneous access (multiusers)
* Provide loading, backup, and recovery

【NOT the function of DBMS: design the database to be used】

**Entity-relationship (ER) diagram**: a graphical data modeling technique that represents the main entities and their corresponding relationships within a system or an enterprise

Basic constructs:

* **Entity**: real-word object distinguishable from other objects (an entity is described using a set of attributes)
  + Each attribute has a domain (e.g. float, integer, string, date)
  + Each entity set has a key

Entity set: the collection of all entities of a particular entity type in the database at any point in time

* **Attributes**
* **Relationship**: association among 2 or more entities

Relationship degree: the number of participating entity types

e.g. Binary relationship – 2 entities, Ternary relationship – 3 entities, N-ary relationship – n entities

**Relationship constraints**: constraints on the relationship type limit the possible combination of entities that may participate in the corresponding relationship set

Kinds of constraints (2):

* Cardinality constraint: for a relationship set specifies the number of relationships in the set that an entity can participate in: one-to-one, one-to-many, many-to-many
* Participation constraints: indicates if all entities participate in relationship: total OR partial participation
* **Weak entities**: can be identified uniquely only by considering the primary key of another (owner) entity
  + Owner entity set and weak entity set must participate in a 1-to-many relationship set (one owner, many weak entities)
  + Weak entity set must have total participation in this identifying relationship set
* **Specialization & generalization** [Extended ER (EER)]

【Subclasses are specializations of superclass & Superclass is generalization of subclasses】

* Attributes of a superclass are inherited by the subclasses
* Subclass can have its own specific attributes/ relationships

Data abstraction: an abstract view of data that excludes many details that are either too complex or not of interest to the users

Data model: a collection of concepts that can be used to describe the structure of a database to achieve data abstraction

* **Schemas**: the metadata or data describing data – static
* **Instance**: the data in the database at a particular time – dynamic

**Relational model**

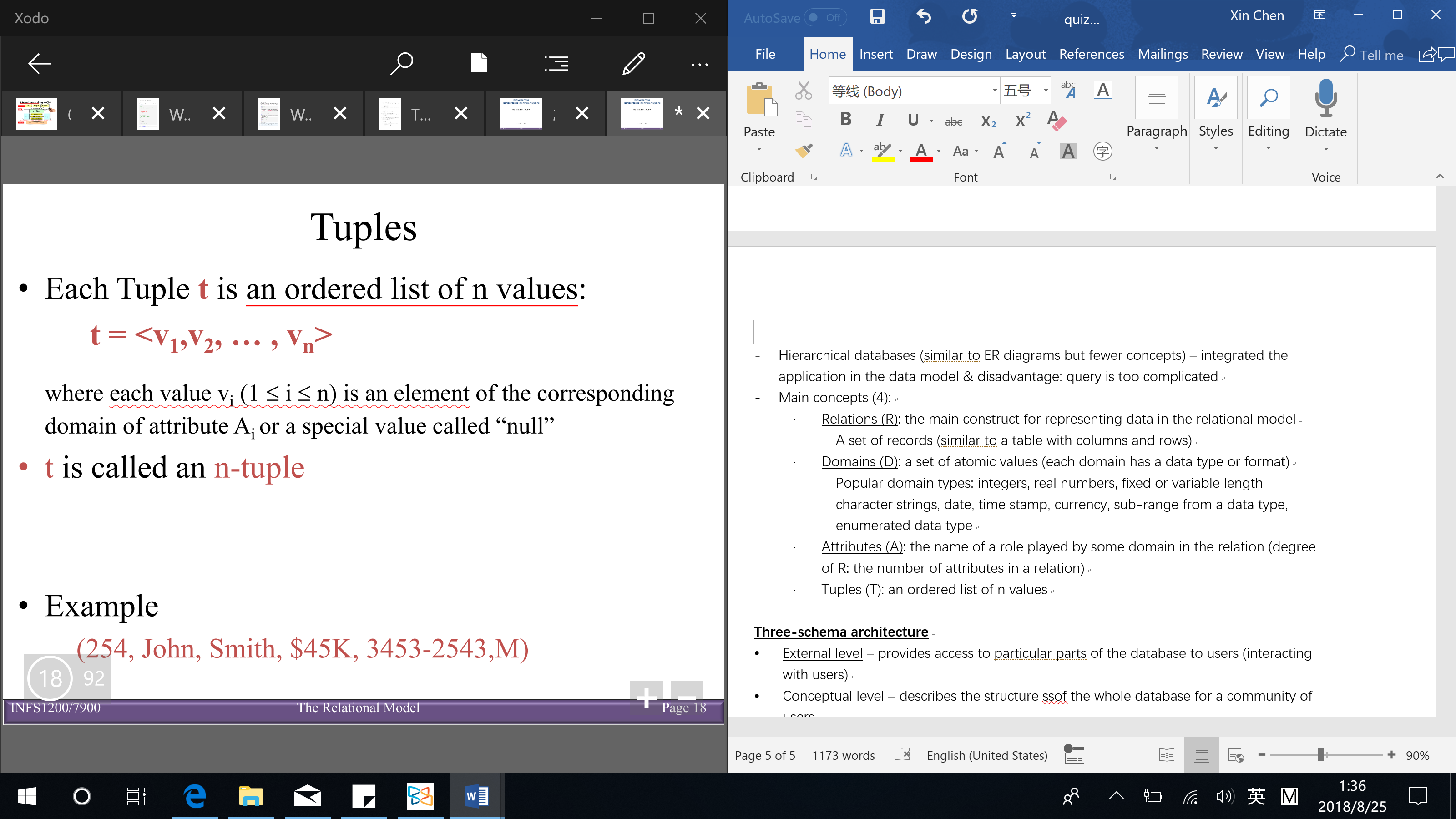
* Main concepts (4):
* Relations (R): the main construct for representing data in the relational model

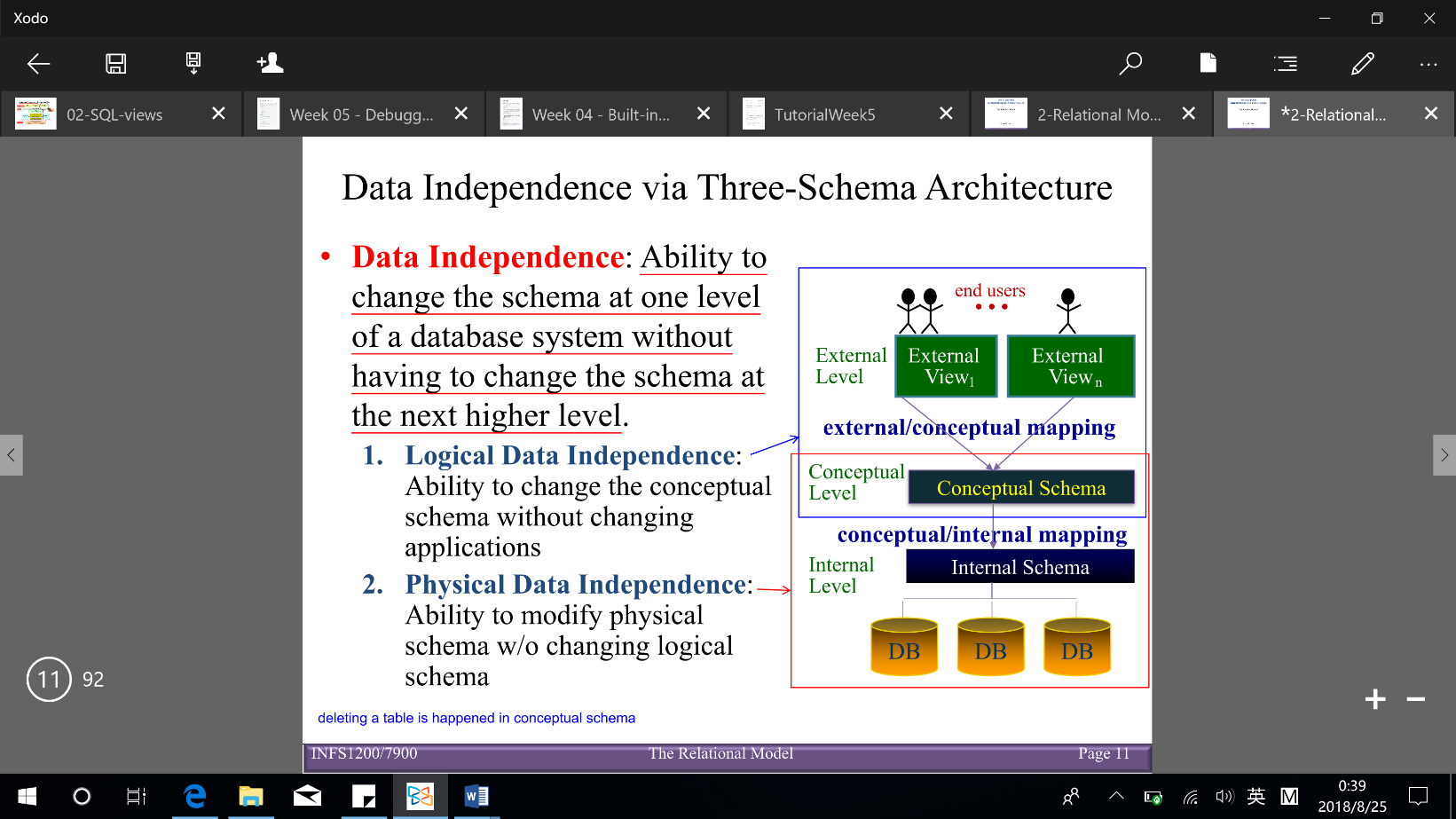
A set of records (similar to a table with columns and rows BUT if it has something merge or combine columns or rows together is a table not a relation)

* Domains (D): a set of atomic values (each domain has a data type or format)

Popular domain types: integers, real numbers, fixed or variable length character strings, date, time stamp, currency, sub-range from a data type, enumerated data type

* Attributes (A): the name of a role played by some domain in the relation (degree of R: the number of attributes in a relation)
* Tuples (T): an ordered list of n values (relations are sets of tuples)





**Three-schema architecture**

* External level – provides access to particular parts of the database to users
* Conceptual level – the structure of the whole database for a community of users
* Internal level – the physical storage structure of the database

**Data independence**: ability to change the schema at one level of a database system without having to change the schema at the next higher level

* Logical data independence: ability to change the conceptual schema without changing applications
* Physical data independence: ability to modify physical schema without changing logical/conceptual schema

Implementation of constraints:

**Integrity constraints (ICs)**: conditions that must be true for any instance of the database

(IC is a statement about all possible instances)

* ICs are specified when schema is defined
* ICs are checked when relations are modified

A legal instance of a relation is one that satisfies all specified ICs

* DBMS should not allow illegal instances
* Avoids data entry errors

Types of ICs: [sequence to check violation of ICs: primary key (Entity, Key), Domain, Referential]

* **Domain constraints**: each attribute in a relation must belong to some domain (right stuff is in the right spot)
* **Key constraints (uniqueness constraint)**

Superkey (SK): subset of attributes that is uniqueness of a relation schema (can have redundant attributes)

(every relation has at least one Superkey – the set of all its attributes: )

Key (K): a minimal Superkey (minimal: removing any attribute means the proposed key is no longer a Superkey)

When have more than one key: Candidate key & Primary key (PK) – only one is selected as PK

* **Entity constraints**: no primary key (PK) can be null
* **Referential integrity constraints** (data exists in the ‘parent’ before the ‘child’)

Specified between 2 relations and are based on the notion of foreign keys

Special situation: self referencing relations

Foreign keys: relate 2 different schemas – a set of attributes FK in relation is a foreign key if: (the attributes of FK have the same domain as the PK attributes of another schema )

Deletion: child table → can just delete & parent table → need to check referential integrity

Modify: opposite of deletion

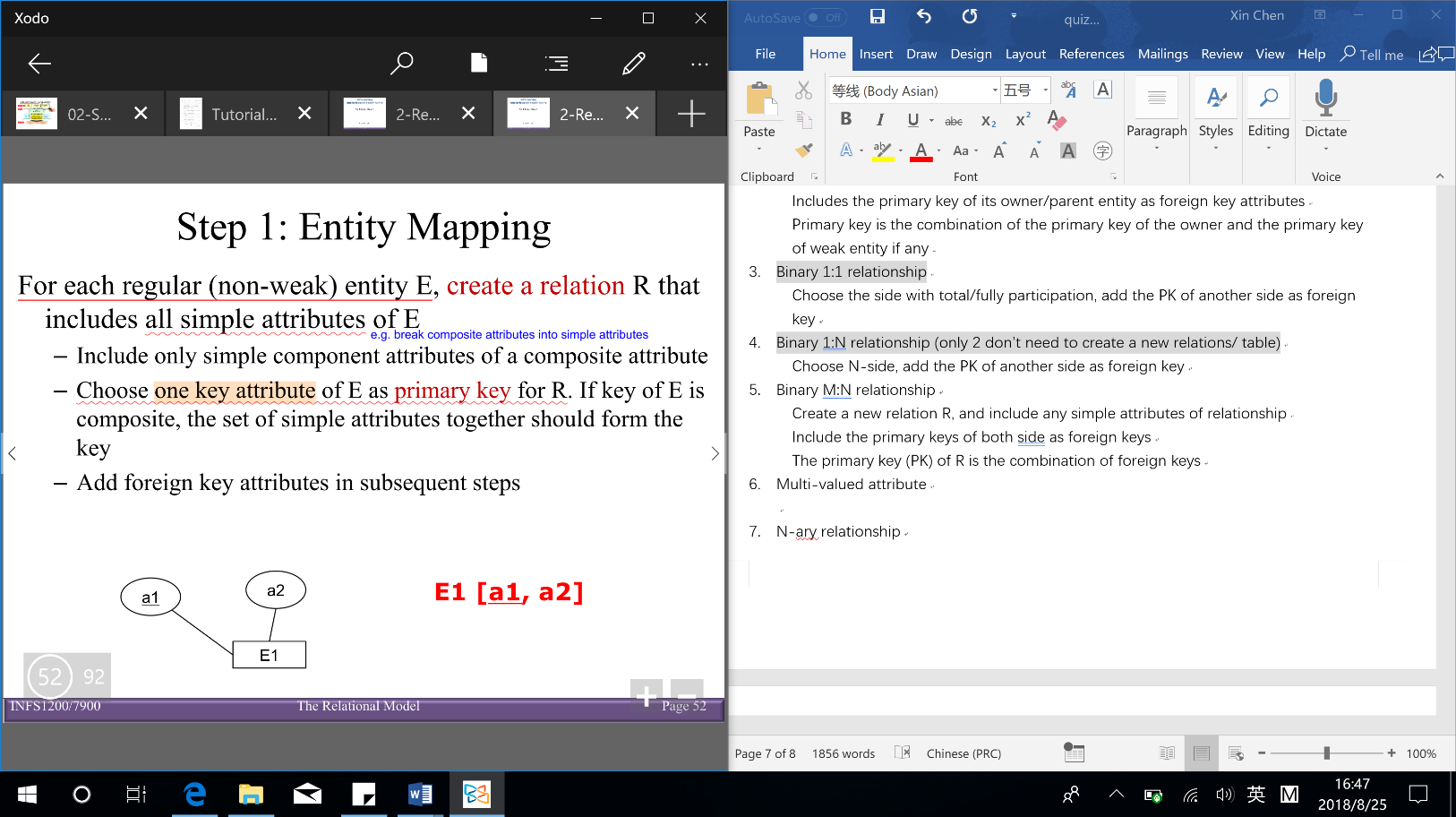
* **User-defined constraints**: general user defined constraints that cannot be enforced by the other constraints

Implemented by using: checks, assertions and triggers

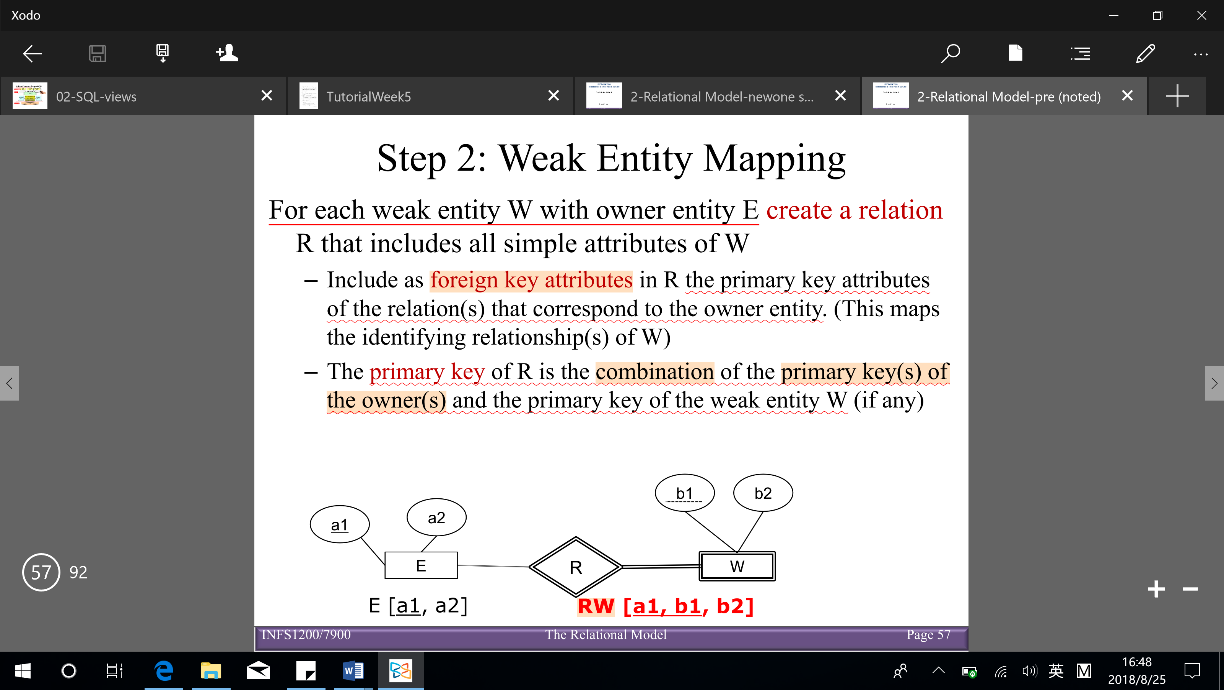
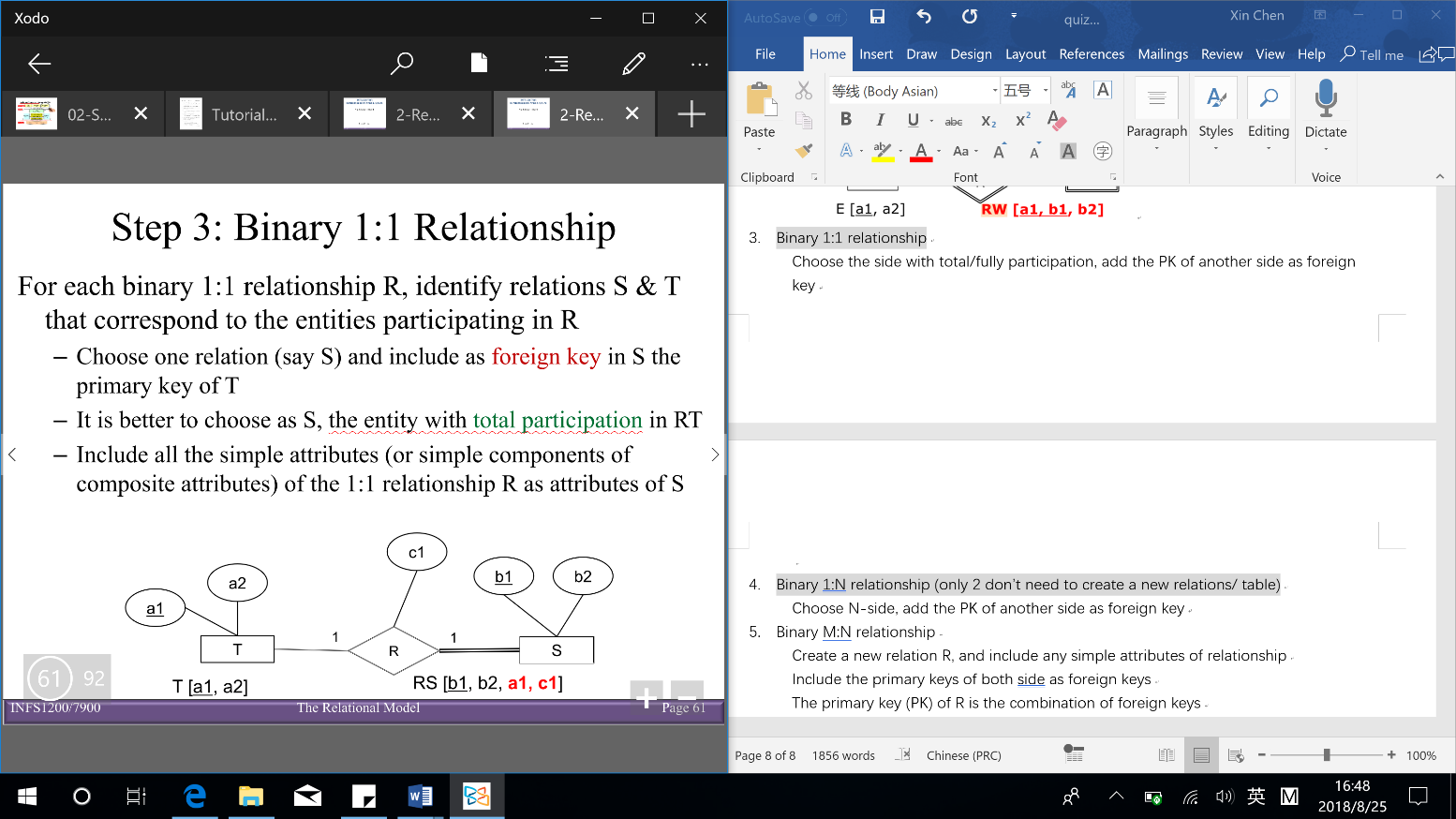
**Mapping ER diagrams to relational models**

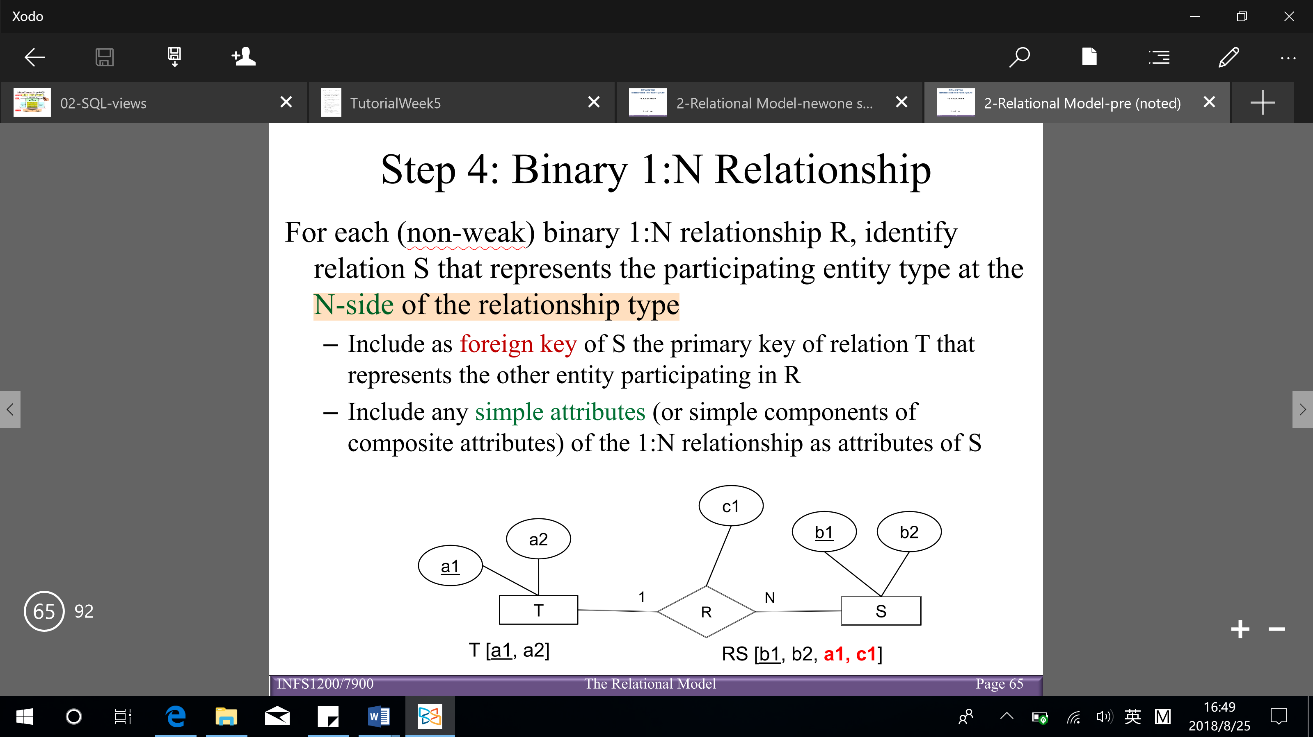
Conceptual perspective:

* Conceptual design – the Entity Relationship (ER)
* Logical design (mapping) – the relational model

Mapping (ER diagram → relational model) steps (8):

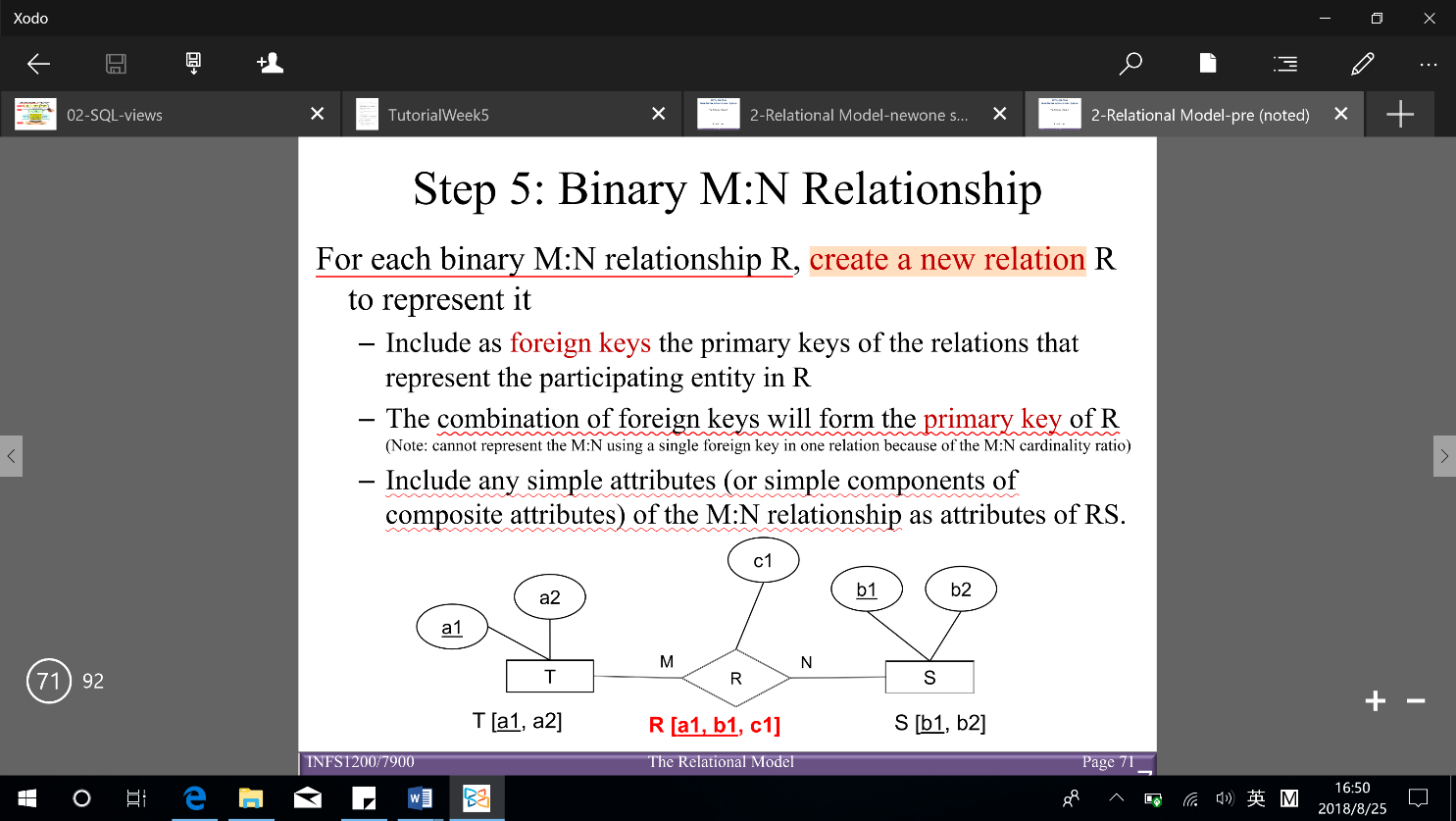
【remember to add foreign key/ reference】

1. Entity – include all simple attributes of a strong entity & Choose one key attribute as primary key (PK)
2. Weak entity – includes all simple attributes of weak entity & Primary key is the combination of the primary key of the owner/parent and the primary key of weak entity (if any)
3. Binary 1:1 relationship – choose the side with total/fully participation, add the PK of another side as foreign key



1. Binary 1:N relationship (only 2 don’t need to create a new relations/ table)

Choose N-side, add the PK of another side as foreign key

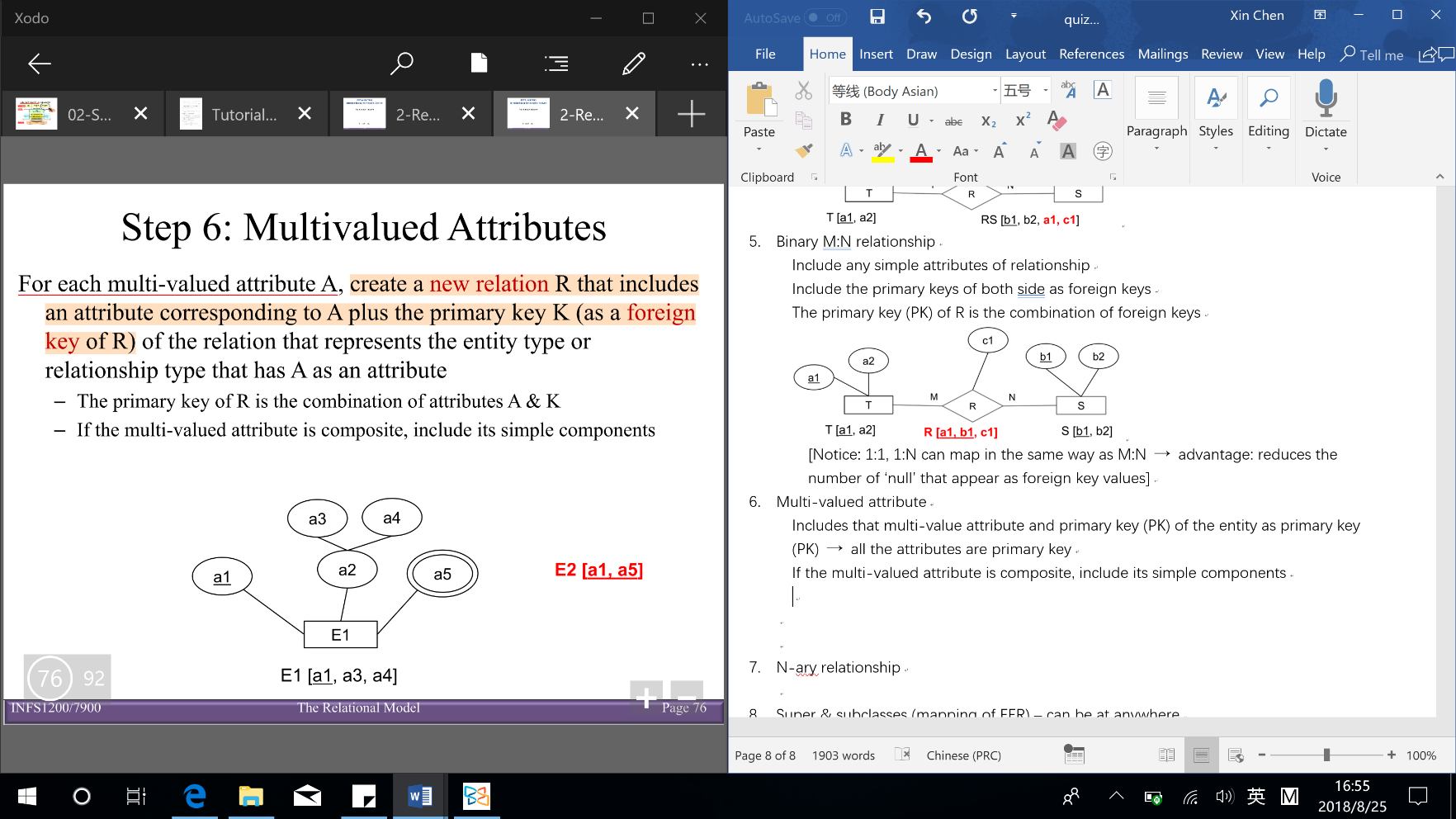
1. Binary M:N relationship

Include any simple attributes of relationship

Include the primary keys of both side as foreign keys

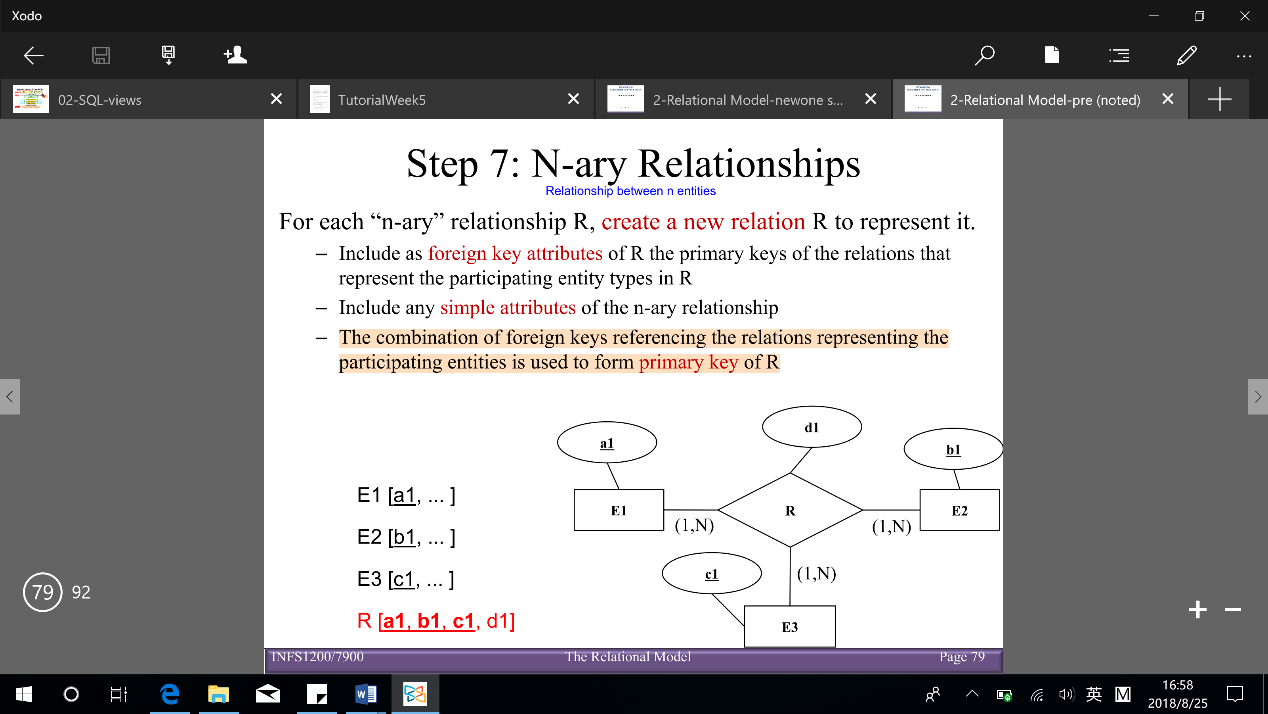
The primary key (PK) of R is the combination of foreign keys

[Notice: 1:1, 1:N can map in the same way as M:N → advantage: reduces the number of ‘null’ that appear as foreign key values]

1. Multi-valued attribute

Includes that multi-value attribute and primary key (PK) of the entity as primary key (PK) → don’t have non-key attribute (ways to distinguish with weak entity: weak entity have other non-key attribute)

If the multi-valued attribute is composite, include its simple components

1. N-ary relationship

Include the primary key of all sides as foreign key

The combination of foreign keys (with many side relationship) is primary key (PK)

Include any simple attributes of n-ary relationship

1. Super & subclasses (mapping of EER) – can be at anywhere

Create a relational table for the superclass

Create a relational table for each subclass

The primary key (PK) of each subclass is the primary key of the superclass (it also is the foreign key)

**Reverse engineering (from schema/ relational model to ER diagram)**:

1. Identify strong entities – the primary key (PK) has no foreign keys referencing
2. Weak entities – if primary key (PK) is combining an existing primary key (via foreign key link) & it has additional non-key attributes
3. 1:1 or 1:N relationships – have foreign key reference that is non-key attribute → 1:1 (1 side) or 1:N (n side) relationship
4. M:N relationships – the primary key (PK) is a combination of existing primary key (via foreign key links)
5. Multi-valued attributes – the primary key combines an existing primary key (via foreign key link) & adds new attributes to the key & have NO additional non-key attributes
6. N-ary relationships – the primary key (PK) is combination of existing of more than 2 primary keys
7. Super and subclasses – A relation/ table has the same primary key (PK) as an entity (via foreign key link)

Informal design guidelines

* Semantics of the attributes

Design each relation so that it’s easy to explain its meaning

Do not combine attributes from multiple entity types and relationship types into a single relation

* Reducing the redundant values in tuples (minimize the storage space) & reducing the null values in tuples
* Disallowing spurious (fake/ false) tuples – when join and decomposition

Modification anomaly: data inconsistency that results from data redundancy

Deletion anomaly: loss of certain attributes because of the deletion of other attributes

Insertion anomaly: lack of ability to insert some attributes without the presence of other attributes (key cannot be null – entity constraint)

Incorrect grouping may cause update anomalies which may result in inconsistent data or even loss of data

**Decomposing a relation** (one table to many tables) – replace the relation R by 2 or more relations such that:

Each new relation contains a subset of the attributes of R (and no attributes not appearing in R)

Every attribute of R appears in at least one new relation

**Join**: is the (natural) join of the 2 relations – each tuple of is concatenated (link together in a chain or series) with every tuple in having the same values on the common attributes

Lossless-join decompositions: decomposition of into and is a lossless-join with respect of a set of functional dependences F if, for every instance r that satisfies F: (If we break a relation R, into bits, when we put the bits back together, we can get the exactly R back again)

Lossy-join decomposition (loss – loss of information OR addition of spurious information)

**Functional dependencies (FD)** – one attribute determines another attribute: holds if for every legal instance, for all tuples : if

(means that given 2 tuples in r, if the x values agree, then y values must also agree)

Key: a minimal set of attributes that functionally determines all the attributes

Superkey: a relation uniquely identifies the relation, but does not have to be minimal

Closure of F: the set of all FDs implied by F (closure for a set of attributes X is denoted by )

|  |  |  |
| --- | --- | --- |
| Left | Both | Right |
| In every key | Might be in some keys | Not in key |

Inference rules:

**Armstrong’s Axioms** ( are sets of attributes):

* Reflexivity:
* Augmentation:
* Transitivity:

**Additional rules**:

* Union:
* Decomposition:

Role of FDs in detecting redundancy (suppose 3 attributes, A, B, C):

* No FDs hold: there is no redundancy here
* Given A → B: several tuples could have the same A value, and if so, they’ll all have the same B value

**Normalization**: the process of removing redundancy from data