COMP 3311
DATABASE MANAGEMENT SYSTEMS

LECTURE 4
RELATIONAL MODEL AND RELATIONAL DATABASE DESIGN

RELATIONAL MODEL & RELATIONAL DATABASE DESIGN: OUTLINE

Relational Model

Reduction of E-R Schemas to Relational Schemas

Functional Dependencies and Normalization

RELATIONAL MODEL

The relational model represents the data for an application as a collection of tables.

Relational Model		Representation	Notation
Relation	⇔	table	$R(A_1, A_2, \ldots, A_n)$
Attribute	⇔	column	A_i
Domain	⇔	type and range of attribute values	$dom(A_i)$
Tuple / Record	⇔	row	
Attribute value	⇔	value in a table cell	

Examples of attribute domains:

Attribute	Domain
age	[0-100]
name	50 alphabetic characters
salary	non-negative integer

RELATIONAL MODEL: SCHEMAS & INSTANCES

- A set of relation schemas define a relational database.

Employee(empId, name, address, hkid, projectNo)

Project(projectNo, name, budget)

relation schema

- A table can be used to show the instances of a relation schema.

Employee

empId	name	address	hkid	projectNo
1	Holmes D.	86 Queen	A450361	3
5	Chan B.	21 Minto	C461378	2
35	Hui J.	16 Peak	F562916	1
8	Bell G.	53 Water	A417394	2
15	Wing R.	58 Aster	C538294	3

Project

projectNo	name	budget
1	E-commerce	200,000
2	Stock control	100,000
3	Web store	500,000

RELATIONAL MODEL: FORMAL DEFINITIONS

- A tuple *t* is an ordered sequence of n values $(v_1, v_2, \ldots, v_n)$ such that $v_i \in dom(A_i)$ or is null.

- The **Cartesian product** over $dom(A_1), dom(A_2), \ldots, dom(A_n)$ is the set of all possible tuples $\{t_1, t_2, \ldots, t_m\}$ with first element $\in dom(A_1)$, second element $\in dom(A_2)$, … and last element $\in dom(A_n)$.

  Example: Let dom(name) = {Lee, Cheung} and dom(grade) = {A, B, C}.

  Then the Cartesian product of the two domains is

  dom(name) × dom(grade) = {<Lee, A>, <Lee, B>, <Lee, C>,
  <Cheung, A>, <Cheung, B>, <Cheung C>}

  ☞ The Cartesian product is not commutative since the domain order matters.

- A relation instance (or simply relation) **r** over relation schema **R** is

  $$\mathbf{r}(\mathbf{R}) \subseteq dom(A_1) \times dom(A_2) \times \ldots \times dom(A_n).$$

  ☞ A relation instance is *any subset* of the Cartesian product of its domains.

  Example: r(**R**(name, grade)) = {<Lee, A>, <Cheung C>} ⊆ dom(name) × dom(grade)

RELATIONAL MODEL: PROPERTIES OF RELATIONS

- Tuples in a relation are *not ordered*, even though they are represented in a tabular form.

  ☞ **Recall that a relation is a set of tuples.**

- All attribute values are *atomic*.

  ☞ Multivalued and composite attribute values are not allowed in relations, although they are permitted in the E-R model.

- The degree of a relation is the number of attributes.

- The cardinality of a relation is the number of tuples.

  ☞ Note that this is not the same as the cardinality constraint in the E-R model.

RELATIONAL MODEL: KEYS

Superkey A superkey, $\mathcal{S}$, of relation **R**={A_1, A_2, …, A_n} is a set of attributes $\mathcal{S} \subseteq$ **R** such that for any two tuples t_1 and $t_2 \in$ **r**(**R**), $t_1[\mathcal{S}] \neq t_2[\mathcal{S}]$.

☞ A superkey is *any* set of attributes that can uniquely identify a tuple in **r**(**R**).

Employee(empId, name, address, hkid, projectNo)
where empId and hkid are unique.

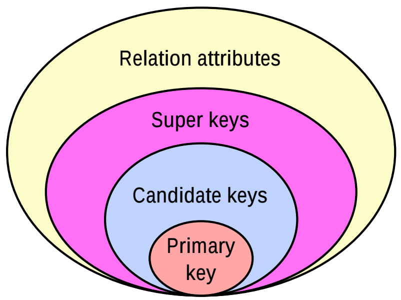
Possible superkeys:
- empId
- hkid
- {empId, name}
- {hkid, name, address}
- plus many others

Note

- In the relational model, every relation is *required* to have a candidate key.
- In a relational DBMS, a relation is *not required* to have a candidate key.

Candidate key A superkey that is minimal. (A relation may have *several candidate keys*.)

Primary key One of the candidate keys. (A relation can have *at most one primary key* selected by the database designer.)



RELATIONAL MODEL: CONSTRAINTS

Entity integrity constraint

If X is a primary key of **R**, then X cannot contain null values.

Referential integrity (foreign key) constraint

S($\underline{k_S}$,…)   **T**(…, fk_S)

Given two relations **S** and **T**, relation **T** may reference relation **S** via a set of attributes fk_S that forms the primary key k_S of **S**.

The attributes fk_S in **T** are called a foreign key.

The value of the foreign key fk_S in a tuple of **T** must either be equal to the value of the primary key k_S of a tuple in **S** or be entirely null.

RELATIONAL MODEL: CONSTRAINTS (CONT'D)

Referential integrity (foreign key) constraint

The attribute projectNo in Employee is a foreign key since it references the primary key projectNo of Project.

Employee(empId, …, projectNo) → Project(projectNo, name, budget)

Employee

empId	name	address	hkid	projectNo
1	Holmes D.	86 Queen	A450361	3
5	Chan B.	21 Minto	C461378	2
35	Hui J.	16 Peak	F562916	1
8	Bell G.	53 Water	A417394	2
15	Wing R.	58 Aster	C538294	3

Project

projectNo	name	budget
1	E-commerce	200,000
2	Stock control	100,000
3	Web store	500,000

Values of projectNo in Employee must be equal to projectNo in Project or be null.

E-R TO RELATION SCHEMA REDUCTION: OVERVIEW

We need to reduce:

generalizations / specializations	⟹	inheritance, coverage
attributes	⟹	composite, multivalued
entities	⟹	strong, weak
relationships	⟹	degree (e.g., unary, binary)
	⟹	cardinality, participation and inclusion constraints

Cardinality/participation constraints in the E-R model reduce to referential integrity constraints in the relational model.

E-R TO RELATION SCHEMA REDUCTION: REFERENTIAL INTEGRITY ACTIONS

S(k_S,…) **T**(…, fk_S)

If relation **T** contains the primary key k_S of relation **S** as a foreign key fk_S, which can be specified as the foreign key constraint

foreign key (fk_S) references **S**(k_S)

then the value of fk_S in a tuple of **T** must either be equal to the value of the primary key k_S of a tuple in **S** or be entirely null.

To enforce this constraint, the following actions are required.

For E-R model: total participation

on delete cascade - Delete all tuples with foreign key values in **T** that match the primary key value of the deleted tuple in **S**.

For E-R model: partial participation

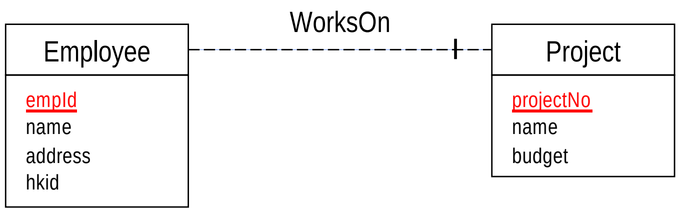
on delete set null - Set to null the foreign key value of all tuples in **T** whose foreign key value matches the primary key value of the deleted tuple in **S**.

E-R TO RELATION SCHEMA REDUCTION: REFERENTIAL INTEGRITY ACTIONS

For total participation ⟹ on delete cascade

Suppose project 3 is deleted.

What changes are required to the Employee relation—cascade or set null?



Employee(empId, ..., projectNo)

Project(projectNo, name, budget)

Employee

empId	name	address	hkid	projectNo
1	Holmes D.	86 Queen	A450361	3
5	Chan B.	21 Minto	C461378	2
35	Hui J.	16 Peak	F562916	1
8	Bell G.	53 Water	A417394	2
15	Wing R.	58 Aster	C538294	3

Project

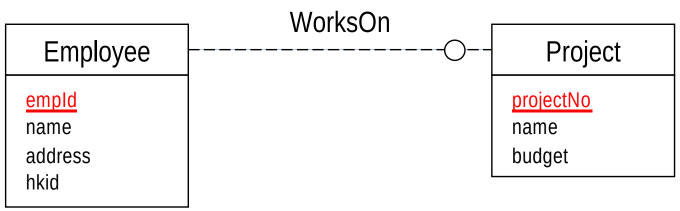
projectNo	name	budget
1	E-commerce	200,000
2	Stock control	100,000
3	Web store	500,000

E-R TO RELATION SCHEMA REDUCTION: REFERENTIAL INTEGRITY ACTIONS

For partial participation ⟹ on delete set null

Suppose project 3 is deleted.

What changes are required to the Employee relation—cascade or set null?



Employee(empId, …, projectNo)

Project(projectNo, name, budget)

Employee

empId	name	address	hkid	projectNo
1	Holmes D.	86 Queen	A450361	3
5	Chan B.	21 Minto	C461378	2
35	Hui J.	16 Peak	F562916	1
8	Bell G.	53 Water	A417394	2
15	Wing R.	58 Aster	C538294	3

Project

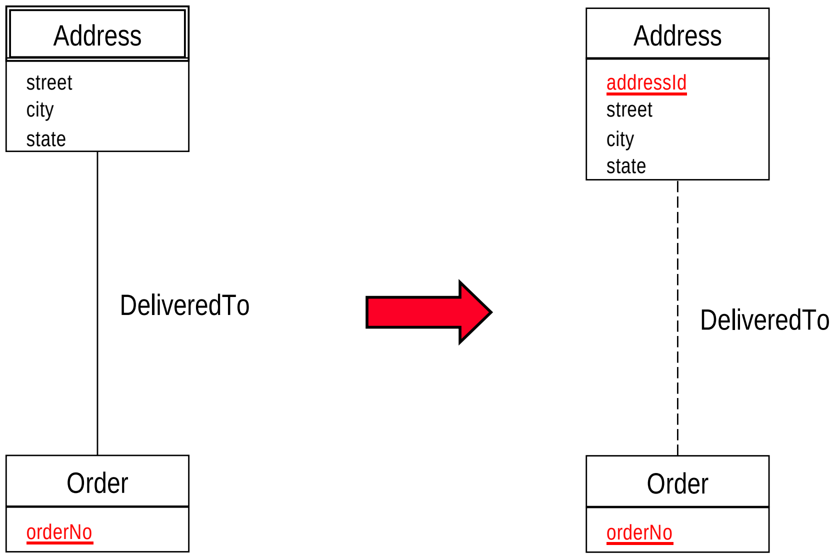
projectNo	name	budget
1	E-commerce	200,000
2	Stock control	100,000
3	Web store	500,000

E-R TO RELATION SCHEMA REDUCTION: SURROGATE KEY

- A surrogate key is a *new attribute* introduced into an entity to be the primary key of the entity.

- Surrogate keys are used to make weak entities strong or to replace a strong entity's key, if it consists of many attributes, to facilitate schema reduction.

☞ **A surrogate key is usually invisible to the user.**

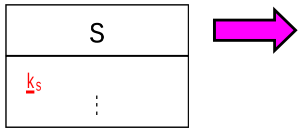


The values of a surrogate key are usually sequentially assigned numbers.

REDUCE STRONG ENTITIES

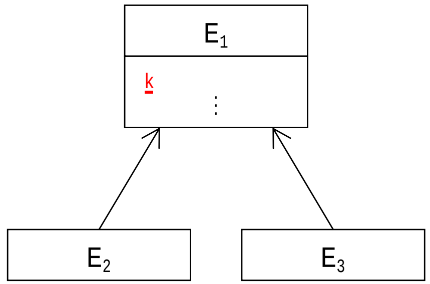
For a strong entity **S**:

- Create a relation schema R_S with all the attributes of entity **S**, *excluding* multivalued and composite attributes.
- The primary key of relation R_S is the primary key of entity **S**.



R_S(<u>k_S</u>, …)

REDUCE GENERALIZATIONS/SPECIALIZATIONS



Option 1: Reduce *all entities* to relation schemas.

– Create a relation schema for each entity (superclass and subclass).

– For each relation schema created for a subclass entity:

Add: 1. the primary key, k, of the superclass entity as a foreign key fk.

➢ The foreign key fk becomes the primary key.

2. a foreign key constraint:

foreign key (fk) references superclass-relation-schema(k).

3. a referential integrity action: on delete cascade.

⇒ E_1(k, …)
E_2(k, …)
E_3(k, …)

Option 2: Reduce *only subclass entities* to relation schemas.

– Create a relation schema for each subclass entity.

⇒ E_2(k, …)
E_3(k, …)

– For each relation schema created for a subclass entity:

Add: 1. all the attributes of the superclass entity.

➢ The primary key is the primary key of the superclass entity.

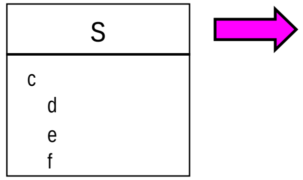
☞ **Should be used only for total, disjoint generalizations/specializations!**

REDUCE COMPOSITE/ MULTIVALUED ATTRIBUTES

For a composite attribute **C** in a strong entity **S**:

Option 1: Create a single attribute by "concatenating" the components of the composite attribute.

Option 2: Create a separate attribute for each component of the composite attribute.

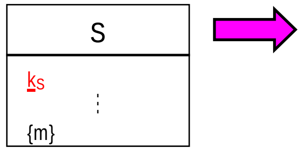


Option 1
S(…, c, …)

Option 2
S(…, d, e, f, …)

For a multivalued attribute **M** in a strong entity **S:**

- Create a relation schema SM with an attribute **a** that corresponds to **M** and attributes, fk_S, corresponding to the primary key of entity **S**.
- The primary key of relation SM is the union of all its attributes.
- Add: 1. a foreign key constraint: foreign key (fk_S) references **S**(k_S).
  2. a referential integrity action: on delete cascade.



SM($\underline{k_S, a}$)

☞ **If a relationship has a composite/multivalued attribute, "convert" the relationship to an entity before applying the reduction.**

REDUCE WEAK ENTITIES

For a weak entity **T** that depends on *only one* strong entity **S**:

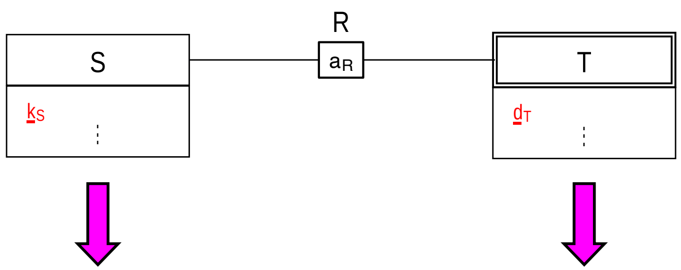
– Create a relation schema R_T with attributes of the weak entity **T**.

– Include attributes a_R of relationship **R** in relation R_T.

– Include as foreign key attributes fk_S in relation R_T, the primary key attributes k_S of the strong entity **S**.

– The primary key of relation R_T is the union of the foreign key attributes fk_S and the *discriminator* d_T , if any, of the weak entity **T**.

– Add: 1. a foreign key constraint: foreign key (fk_S) references **S**(k_S).
2. a referential integrity action: on delete cascade.



R_S($\underline{k_S}$, …)
(*previously reduced*)

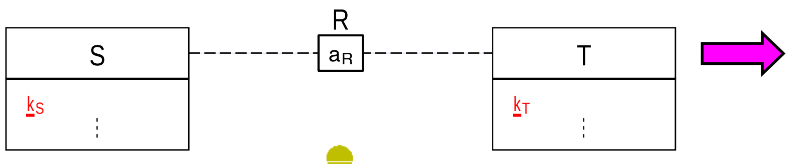
R_T($\underline{k_S, d_T}$, a_R, …)

REDUCE RELATIONSHIPS

For each binary relationship **R**:

- Create a new relation schema R_R.
- Include as foreign key attributes in relation R_R the primary keys of the entities related by relationship **R**.
- Include attributes a_R of relationship **R**, if any, as attributes of relation R_R.
- The primary key of relation R_R is
  - *binary 1:1 relationship* $\Longrightarrow$ the primary key of either entity.
  - *binary 1:N relationship* $\Longrightarrow$ the primary key of the entity on the N-side of the relationship.
  - *binary N:M relationship* $\Longrightarrow$ the union of the primary keys of the two participating entities.
- Add: a foreign key constraint for each foreign key with the referential integrity action on delete cascade.

☞ **This reduction minimizes null values.**



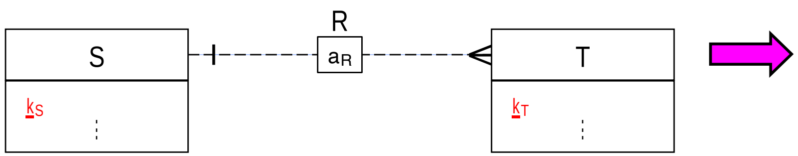
$R_R(k_S, k_T, a_R, \ldots)$

SCHEMA COMBINATION

- For 1:1 relationships, the relation schema for the relationship can be combined with the relation schema for either entity.

- For 1:N relationships, the relation schema for the relationship can be combined with the relation schema for the entity on the N-side.

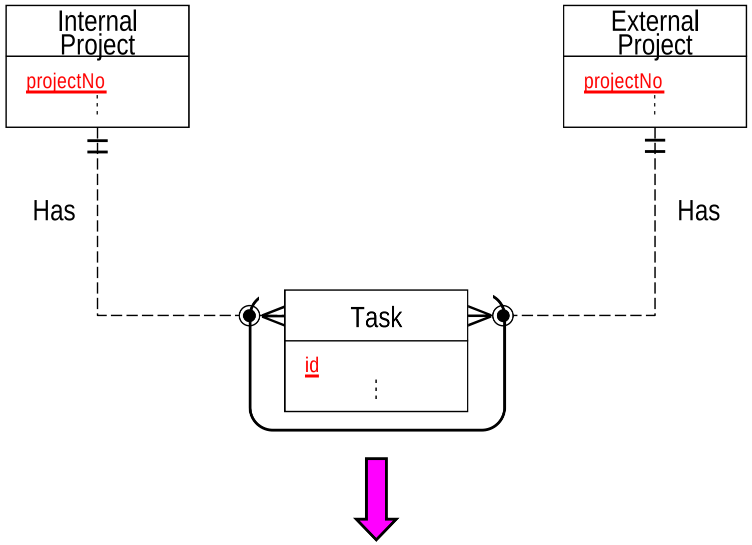
Add: 1. a foreign key constraint for the foreign key.
2. a referential integrity action for the foreign key that is determined by the participation constraint of the entity into which the foreign key is placed.
   - partial: on delete set null
   - total: on delete cascade



$R_T(\underline{k_T}, k_S, a_R, \ldots)$

REDUCE EXCLUSION CONSTRAINTS

Option 1



Task(id, …, *externalProjNo*, *internalProjNo*)
foreign key (externalProjNo) references ExternalProject(projectNo)
on delete cascade
foreign key (internalProjNo) references InternalProject(projectNo)
on delete cascade
check ((externalProjNo is not null and internalProjNo is null) or
(internalProjNo is not null and externalProjNo is null))

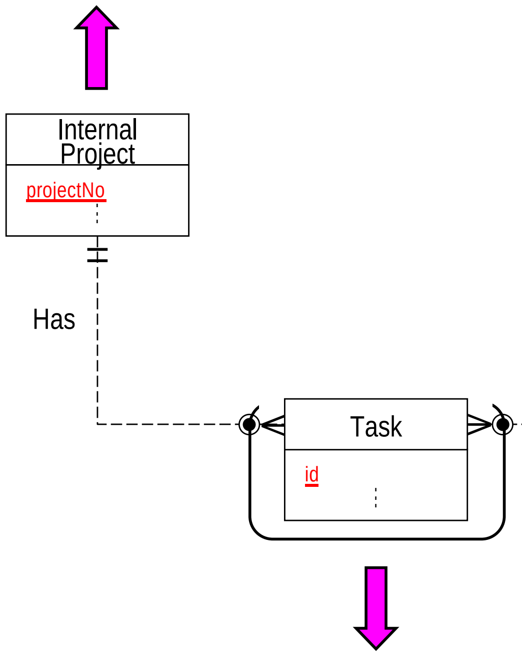
An additional constraint is needed to ensure that only one of externalProjNo and internalProjNo has a value.

REDUCE EXCLUSION CONSTRAINTS (CONT'D)

Option 2

Can only be used if the constrained entity types have the same key.

InternalProject(projectNo, …) ExternalProject(projectNo, …)



Task(id, …, *projectNo*, projectType)

Attribute projectType identifies whether the task is related to an InternalProject or an ExternalProject instance.

☞ **Cannot specify referential integrity constraints or actions for Task!**

BUT the exclusion constraint is automatically enforced.