Relationship Set Keys

- General rule for binary relationships
  - one-to-one: primary key of either entity set
  - one-to-many: primary key of the many side
  - many-to-many: union of primary keys of the associate entity sets

- n-ary relationships
  - More complicated rules
Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints.

### Diagram

```
  instructor
     ID
     name
     salary
  -------
     0..*
  advisor
     1..1
  -------

  student
     ID
     name
     tot_cred
```

#### Summary: Keys for Relationship Sets

- Primary keys of entity sets together form super key of a relationship set.
  - (s_id, i_id) is the super key of advisor
  - **NOTE:** this means an entity pair have at most one relationship in a relationship set.
    - Example: if we wish to track multiple meeting dates between a student and her advisor, we cannot assume a relationship for each meeting.
    - Fix: use a multivalued attribute.
- Must consider the mapping cardinality of the relationship set when identifying candidate keys.
- Consider relationship set semantics in selecting the primary key if more than one candidate key.
Weak Entity Sets

- An entity set that does not have a primary key is referred to as a **weak entity set**.
- The existence of a weak entity set depends on the existence of an **identifying entity set**
  - It must relate to the identifying entity set via a one-to-many relationship set from the identifying to the weak entity set
  - Weak side of relationship set must be *total*
  - **Identifying relationship** depicted using a **double diamond**
- The **discriminator** *(or partial key)* of a weak entity set is the set of attributes that *help* distinguish among all the entities of a weak entity set.
- Primary key of a weak entity set is:
  - primary key of the strong entity set on which the weak entity set is existent-dependent,
  - plus the weak entity set’s discriminator.

Weak Entity Sets (Cont.)

- We underline the discriminator of a weak entity set with a dashed line.
- We put the identifying relationship of a weak entity in a double diamond.
- Primary key for *section* – *(course_id, sec_id, semester, year)*
Weak Entity Sets (Cont.)

- Note: the primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.

- If course_id were explicitly stored, section could be made a strong entity, but then the relationship between section and course would be duplicated by an implicit relationship defined by the attribute course_id common to course and section.

- Duplication is bad.
Symbols Used in E-R Notation (Cont.)

- Many-to-many relationship
- Many-to-one relationship
- One-to-one relationship
- Cardinality limits
- Role indicator
- Role name

E-R Diagram for a University Enterprise
Reduction to Relational Schemas

Outline

- Relational Algebra (6.1)
- E/R Model (7.2 - 7.4)
- E/R Diagrams (7.5)
- Reduction to Schema (7.6)
- Relational Database Design (7.7)
- Functional Dependencies (8.1 – 8.4)
- Normalization (8.5 – 8.7)
Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of schemas.
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set.
- Each schema has a number of columns (generally corresponding to attributes), which have unique names.

Representing Entity Sets

- A strong entity set reduces to a schema with the same attributes:
  \[ \textit{student}(\textit{course\_ID}, \textit{title}, \textit{credits}) \]
- A weak entity set becomes a table that includes a foreign key for the primary key of the identifying strong entity set:
  \[ \textit{section} (\textit{course\_id}, \textit{sec\_id}, \textit{semester}, \textit{year}) \]
Representing Relationship Sets

- A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- Example: schema for relationship set advisor:

  \[
  \text{advisor}(s\_id, i\_id)
  \]

Redundancy of Schemas

- Many-to-one and one-to-many relationship sets that are \textbf{total} on the many-side can be represented by adding an extra attribute(s) to the “many” side, containing the primary key of the “one” side.
- Example:
  - get rid of \textit{inst_dept} by:

  \[
  \text{instructor}(ID, \text{name}, \text{salary}) \rightarrow \text{instructor}(ID, \text{dept_name}, \text{name}, \text{salary})
  \]
Redundancy of Schemas (Cont.)

- For one-to-one relationship sets, either side can be chosen to act as the “many” side
  - That is, extra attribute can be added to either of the tables corresponding to the two entity sets (both have to be total)

  ![Diagram of Instructor and Department tables]

  instructor(ID, dept_name, name, salary)

  or

  department(dept_name, ID, building, budget)

Redundancy of Schemas (Cont.)

- If participation is partial on the “many” side, replacing a relationship schema by an extra attribute in the schema corresponding to the “many” side could result in null values (generally avoided)
  - i.e. the approach in the previous slides does not work
  - need to represent relationship as a separate table

- A relationship set linking a weak entity set to its identifying strong entity set is redundant.
  - Example: The section schema already contains the attributes that would appear in the sec_course schema
  - Unless otherwise instructed, assume we wish to avoid NULLs when converting to relations, i.e. remove redundant relationship schemas only when total participation on side where adding attribute.
**Composite Attributes**

- Composite attributes flattened out
  - Example: given entity set `instructor`
    - with composite attribute `name`
    - with component attributes `first_name` and `last_name`
    - replace with `name_first_name` and `name_last_name`
      - Prefix omitted if there is no ambiguity
  - Ignoring multivalued attributes, extended instructor schema is
    - `instructor(ID, first_name, middle_initial, last_name, street_number, street_name, apt_number, city, state, zip_code, date_of_birth)`

**Multivalued Attributes**

- Multivalued attribute $M$ of entity $E$ represented by a separate schema $EM$
  - Schema $EM$ includes $E$’s primary key and attribute corresponding to $M$
  - Example: Multivalued attribute `phone_number` of `instructor`:

```
  inst_phone = (ID, phone_number)
```

- Each value of the multivalued attribute maps to separate tuple of $EM$:
  - `instructor` entity with primary key 22222 and numbers 456-7890 and 123-4567 maps to:
    - (22222, 456-7890)
    - (22222, 123-4567)
Reduction to Relational Schemas

Outline

- Relational Algebra (6.1)
- E/R Model (7.2 - 7.4)
- E/R Diagrams (7.5)
- Reduction to Schema (7.6)
- Relational Database Design (7.7)
- Functional Dependencies (8.1 – 8.4)
- Normalization (8.5 – 8.7)
ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now

- Schema per relationship set

 lots of foreign key dependences (weak, relationships..)

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now

- Schema per relationship set

 lots of foreign key dependences (weak, relationships..)
ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set

lots of foreign key dependences (weak, relationships..)
ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set
  - department(dept_name, building, budget)
  - instructor(i_id, dept_name, name, salary)
  - course(course_id, title, credits)
  - section(sec_id, course_id, semester, year)
  - student(i_id, name, tot_cred)
  - classroom(building, room_number, capacity)
  - time_slot(time_slot_id, day, start_time, end_time)

  inst_dept(id, dept_name)
  stud_dept(id, dept_name)
  teaches(teacher_id, sec_id, semester, course_id, year)
  takes(student_id, course_id, sec_id, semester, year, grade)
  advisor(i_id, s_id)
  course_dept(course_id, dept_name)
  sec_time_slot(course_id, sec_id, semester, year, time_slot_id)
  sec_course(course_id, sec_id, semester, year)
  prereq(course_id, prereq_id)
  sec_class(course_id, sec_id, semester, year, building, room_number)

lots of foreign key dependences (weak, relationships..)

ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set
  - department(dept_name, building, budget)
  - instructor(i_id, dept_name, name, salary)
  - course(course_id, title, credits)
  - section(sec_id, course_id, semester, year)
  - student(i_id, dept_name, name, tot_cred)
  - classroom(building, room_number, capacity)
  - time_slot(time_slot_id, day, start_time, end_time)

  inst_dept(id, dept_name)
  stud_dept(id, dept_name)
  teaches(teacher_id, sec_id, semester, course_id, year)
  takes(student_id, course_id, sec_id, semester, year, grade)
  advisor(i_id, s_id)
  course_dept(course_id, dept_name)
  sec_time_slot(course_id, sec_id, semester, year, time_slot_id)
  sec_course(course_id, sec_id, semester, year)
  prereq(course_id, prereq_id)
  sec_class(course_id, sec_id, semester, year, building, room_number)

lots of foreign key dependences (weak, relationships..)
ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set

lots of foreign key dependences (weak, relationships..)
ER Diagram to Relational Schema

- Schema per entity set
  - expand composite attributes
  - new schema for multi-valued
  - drop derived attributes for now
- Schema per relationship set

```
• department(dept_name, building, budget)
• instructor(id, dept_name, name, salary)
• course(course_id, title, credits, dept_name)
• section(sec_id, course_id, semester, year, building, room_number, time_slot_id)
• student(id, dept_name, name, tot_cred)
• classroom(building, room_number, capacity)
• time_slot(time_slot_id, day, start_time, end_time)

• inst_dept(id, dept_name)
• stud_dept(id, dept_name)
• teaches(id, sec_id, semester, course_id, year)
• takes(id, course_id, sec_id, semester, year, grade)
• advisor(i_id, s_id)
• course_dept(course_id, dept_name)
• sec_time_slot(course_id, sec_id, semester, year, time_slot_id)
• sec_course(course_id, sec_id, semester, year)
• prereq(course_id, prereq_id)
• sec_class(building, room_number, capacity, building, room_number)
```

lots of foreign key dependences (weak, relationships..)

---

Binary Vs. Non-Binary Relationships

- Some relationships that appear to be non-binary may be better represented using binary relationships

  - E.g., A ternary relationship `parents`, relating a child to his/her parent1 and parent2, is best replaced by two binary relationships, `parent1` and `parent2`
    - Using two binary relationships allows partial information (e.g., only parent2 being known)
  - But there are some relationships that are naturally non-binary
    - Example: `proj_group`, with several project members
Converting Non-Binary Relationships to Binary Form

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
  - Replace $R$ between entity sets $A$, $B$ and $C$ by an entity set $E$, and $R_A$, $R_B$, $R_C$, relating $E$ with $A$, $B$, and $C$
  - Create a special identifying attribute for $E$
  - Attributes $E$ are all those of $R$, plus special identifier
  - For each relationship $(a_i, b_i, c_i)$ in $R$
    - create a new entity $e_i$ in the entity set $E$
    - add $(e_i, a_i)$ to $R_A$, etc.

\[ (a) \quad (b) \]

Design Considerations

- **Binary versus n-ary relationship sets**
  Although it is possible to replace any nonbinary ($n$-ary, for $n > 2$) relationship set by a number of distinct binary relationship sets, a $n$-ary relationship set shows more clearly that several entities participate in a single relationship.

- **Placement of relationship attributes**
  e.g., attribute *date* as attribute of *advisor* or as attribute of *student*