Module 8: Logical Mapping

Fall 2026

Cheriton School of Computer Science

CS 338: Intro to Database Management

Outline

ER Diagrams to Relational Schemata

Logical Mapping

The *logical mapping* problem is the problem of obtaining a logical design for a database given a conceptual design as input.

We consider the problem when the logical design must be a relational database schema and the conceptual design is an ER diagram.

Main ideas:

- Each entity set maps to a new table.
- Each attribute maps to a new table column.
- Each relationship set maps to either new table columns for existing tables of entity sets or to a new table.

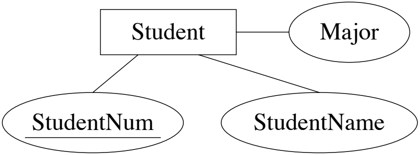
Representing Entity Sets

Entity set E with attributes $a_1, \ldots, a_n$ maps to a new table E with attributes $a_1, \ldots, a_n$.

$\Rightarrow$ (entity of type E) $\leftrightarrow$ (a row in table E)

$\Rightarrow$ (primary key of entity set) $\rightarrow$ (primary key of table E)

Example:



$\updownarrow$

Student

<u>StudentNum</u>	StudentName	Major

Representing Weak Entity Sets

Weak entity set E maps to a new table E.

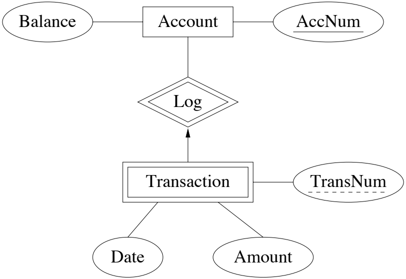
Columns of table E should include:

1. attributes of the weak entity set,
2. attributes of the identifying relationship set, and
3. primary key attributes of entity set for dominating entities.

$\Rightarrow$ (primary key of weak entity set) $\rightarrow$ (primary key of table E)

Representing Weak Entity Sets (cont.)

Example:



$\leftrightarrow$

Account

AccNum	Balance

Transaction

TransNum	AccNum	Date	Amount

Constraint on Transaction:

- `foreign key ( ` AccNum ` ) references` Account.

Representing Relationship Sets

A relationship set may map to either new table columns for existing tables of entity sets or to a new table.

- If the relationship set is an identifying relationship set for a weak entity set then no action needed.
- If we can deduce the general cardinality constraint (1,1) for a component entity set E then add following columns to table E:
  1. attributes of the relationship set, and
  2. primary key attributes of remaining component entity sets.
- Otherwise, relationship set R translates to a new table R.

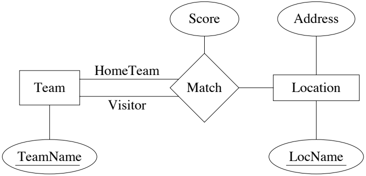
Representing Relationship Sets (cont.)

- Columns of table *R* should include:
  1. attributes of the relationship set, and
  2. primary key attributes of each component entity set.

- Primary key of table *R* determined as follows:
  - If we can deduce the general cardinality constraint (0,1) for a component entity set *E*, then take the primary key attributes for *E*.
  - Otherwise, choose primary key attributes of each component entity.

Representing Relationship Sets (cont.)

Example:



$\leftrightarrow$

Team

<u>TeamName</u>

Location

<u>LocName</u>	Address

Match

<u>HomeTeamName</u>	<u>VisitorTeamName</u>	<u>LocName</u>	Score

Role names and component entity set names are combined to form more descriptive attribute names.

Constraints on table Match:

- `foreign key (` HomeTeamName `) references` Team.
- `foreign key (` VisitorTeamName `) references` Team.
- `foreign key (` LocName `) references` Location.

Outline

Representing Aggregation

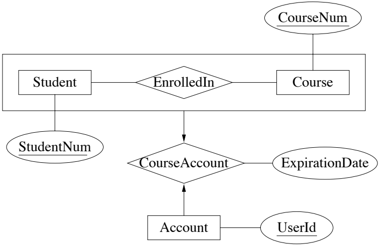
- Always map a relationship set R that is aggregated to a new table R.

  $\Rightarrow$ (*tabular representation of aggregation of R*)
  $= $ (*tabular representation for relationship set R*)

- To represent a relationship set involving the aggregation of R, treat the aggregation like an entity set whose primary key is the *primary key* of the table for R.

Representing Aggregation (cont.)

Example:



$\leftrightarrow$

Student

StudentNum

Course

CourseNum

Account

UserId

EnrolledIn

StudentNum	CourseNum

CourseAccount

UserId	StudentNum	CourseNum	ExpirationDate

Constraints on table EnrolledIn:

- `foreign key (` StudentNum `) references` Student.
- `foreign key (` CourseNum `) references` Course.

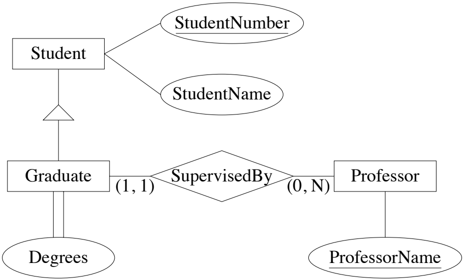
Constraints on table CourseAccount:

- `foreign key (` UserId `) references` Account.
- `foreign key (` StudentNum, CourseNum `) references` EnrolledIn.

Representing Specialization

- Treat an entity set which is a specialization of one or more other entity sets as a weak entity set with an empty discriminator set and that is existence dependent on each of the other entity sets.

Example (also illustrating how to map multi-valued attributes):



$\leftrightarrow$

Student

StudentNumber	StudentName

Graduate

StudentNumber	ProfessorName

Degree

StudentNumber	Degree

Professor

ProfessorName

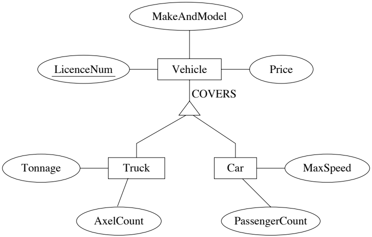
Contraints:

- (Graduate) `foreign key ( StudentNumber ) references` Student.
- (Graduate) `foreign key ( ProfessorName ) references` Professor.
- (Degree) `foreign key ( StudentNumber ) references` Graduate.

Representing Generalization

- Treat a generalization of *n* entity sets as *n* specializations, and add additional constraints for coverage and, if required, for disjointness.

Example:



$\leftrightarrow$

Vehicle

<u>LicenceNum</u>	MakeAndModel	Price

Truck

<u>LicenceNum</u>	Tonnage	AxelCount

Car

<u>LicenceNum</u>	MaxSpeed	PassengerCount

Contraints (expressing the last two in SQL is an exercise):

- (Truck) `foreign key (` LicenceNum `) references` Vehicle.
- (Car) `foreign key (` LicenceNum `) references` Vehicle.
- An assertion that vehicle licence numbers are also truck or car licence numbers.
- An assertion that truck licence numbers are disjoint from car licence numbers (unless `OVERLAPS` annotates the generalization).

Representing Specialization and Generalization with Views

Sometimes an entity set can be mapped to a view instead of a table.[1]

- An entity set E that is a specialization of *one* parent entity set and that satisfies the following conditions qualifies:
  1. only *typing* attributes are declared on E,
  2. no foreign key constraints reference E, and
  3. all entity sets that are specializations of E are also mapped to views.

  Need to add a new two-valued typing attribute `is-`E together with E's existing typing attributes to the parent entity set to enable a view definition for the mapping of E.

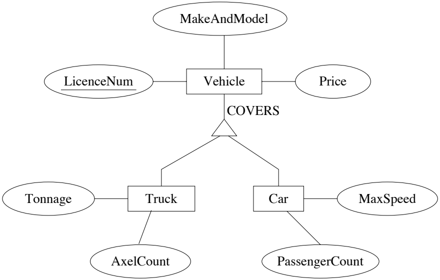
  Multiple typing attributes can be replaced with a single typing attribute when underlying entity sets are disjoint.

- An entity set E that is a generalization of two or more other *child* entity sets and that has no foreign key constraints that reference E qualifies.

  Need to ensure all attributes on E are defined on each child entity set to enable a view definition for the mapping of E.

[1] Increases efficiency with most deployed technology, and improves transparency.

Example: Generalization as Views



↕

Truck

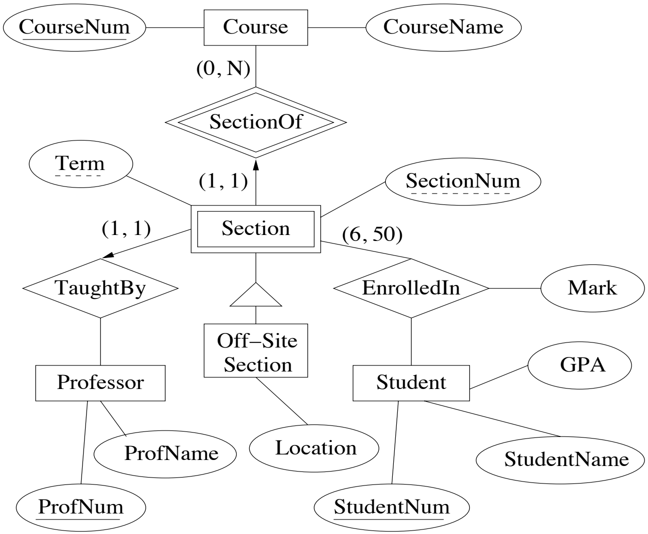
<u>LicenceNum</u>	MakeAndModel	Price	Tonnage	AxelCount

Car

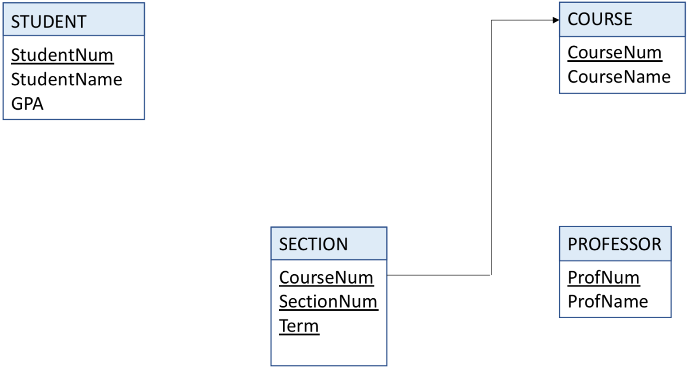
<u>LicenceNum</u>	MakeAndModel	Price	MaxSpeed	PassengerCount

```
create view Vehicle as (
   ( select LicenceNum, MakeAndModel, Price from Truck ) union
   ( select LicenceNum, MakeAndModel, Price from Car ) )
```

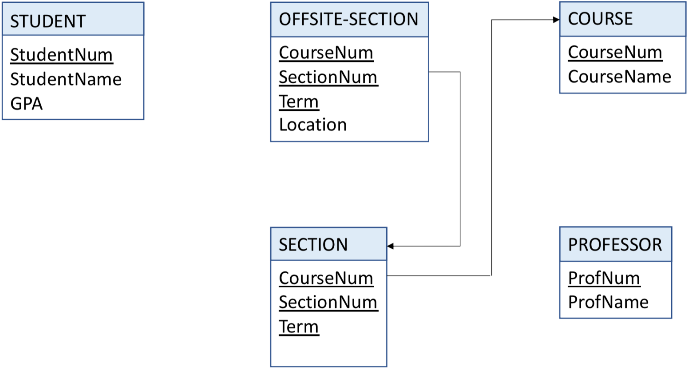
Example Translation: ER Diagram



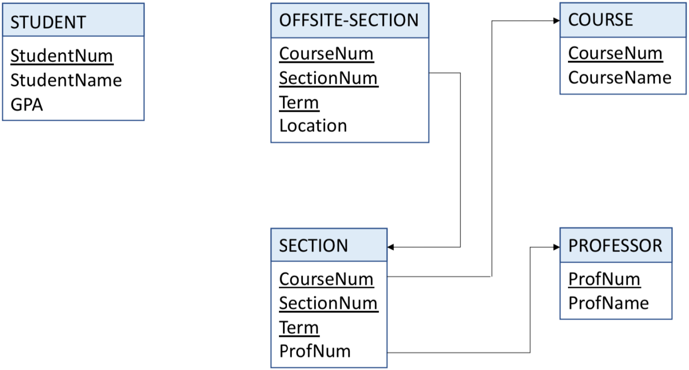
Example Translation: Mapping Entity Sets



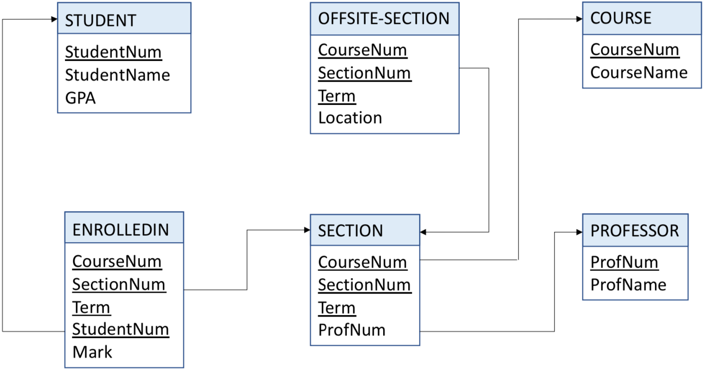
Example Translation: Mapping Specialization



Example Translation: Mapping Relationship Sets



Example Translation: Mapping Relationship Sets (cont'd)



Outline

Triple Store

All triple store relational schemas have an additional view

```
create view TRIPLE as ( <query> )
```

with schema `TRIPLE/(subject,property,object)`.

- Before a mapping is applied, an ER diagram is modified as follows:
  1. make all weak entity sets into regular entity sets; and
  2. for each entity set, add an `OID` attribute and make this the primary key.
- `<query>` is a union of queries `QTi` over each of the other tables `Ti`.
- Assuming `Ti` has schema `Ti/(OID,A1,...,An)`, query `QTi` is defined as follows:

```
(select OID as subject, 'in' as property, 'Ti' as object from Ti)
  union
(select OID as subject, 'A1' as property, A1 as object from Ti)
  union
    :
  union
(select OID as subject, 'An' as property, An as object from Ti)
```

Triple Store (cont'd)

Some observations:

- All queries over a triple store schema have easily obtained equivalent formulations that only mention table `TRIPLE`.

  $\Rightarrow$ (*data in* `TRIPLE`) replicates (*data in all other tables*)

- A relational schema with a single `TRIPLE` table can replicate *any* relational database, and never requires revision.

- Replacing attribute `OID` with attibute `URI` (short for *universal resource identifier*) roughly obtains an RDF (short for *resource description framework*) encoding of data.

- Viewing each value occurring in either column `subject` or column `object` as a graph node and each tuple in `TRIPLE` as a labelled graph edge is the basis of all *graphical data models*.