Relational Database Concepts

Converting an ERD to Relational Tables

PRESENTED BY IMGARD EKOKOBE

Objectives

- Understand the components of ERD diagrams.
- Learn the rules for converting attributes, entity types and relationships to tables and their relationships.

Conversion Rules

- Conversion from an ERD to a table design is important because of industry practice.
- Computer-aided software engineering (CASE) tools support varying notations for ERDs and some of these tools are used support the development of ERDs.
- Since most commercial DBMSs use the Relational Model, you must convert an ERD to relational tables to implement your database design.
- CASE tools are usually used to perform these conversion but having a good understanding how the convention process takes place will help you stand out from your peers.
- Understanding the conversion rules improves your understanding of the entity relationship model, and helps you avoid the common error of using foreign keys in an ERD.
- The main difference between the entity relationship model and the relational model is the named relationships versus foreign keys

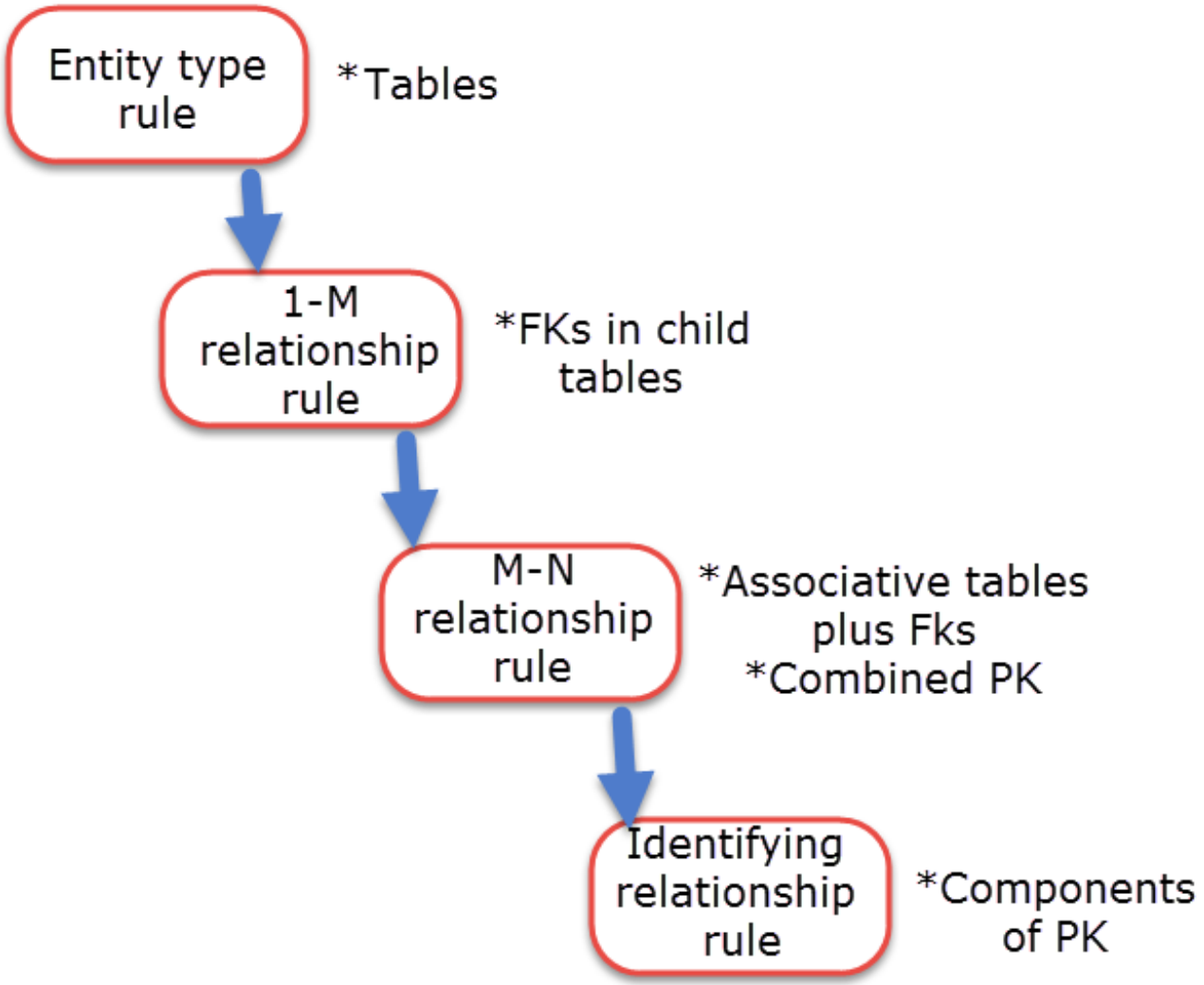
Rules For Converting entity types, relationships, and attributes

There are 4 Conversion Rules:

- Entity Type Rule.
- 1-M Relationship Rule.
- M-N Relationship Rule.
- Identifying Relationship Rule.

Note: You will learn how be able to apply each rule to an ERD and use the rules in the specified order.

Rules For Converting entity types, relationships, and attributes



Rule 1 - Entity Type Conversion Rule

Entity Type Rule:

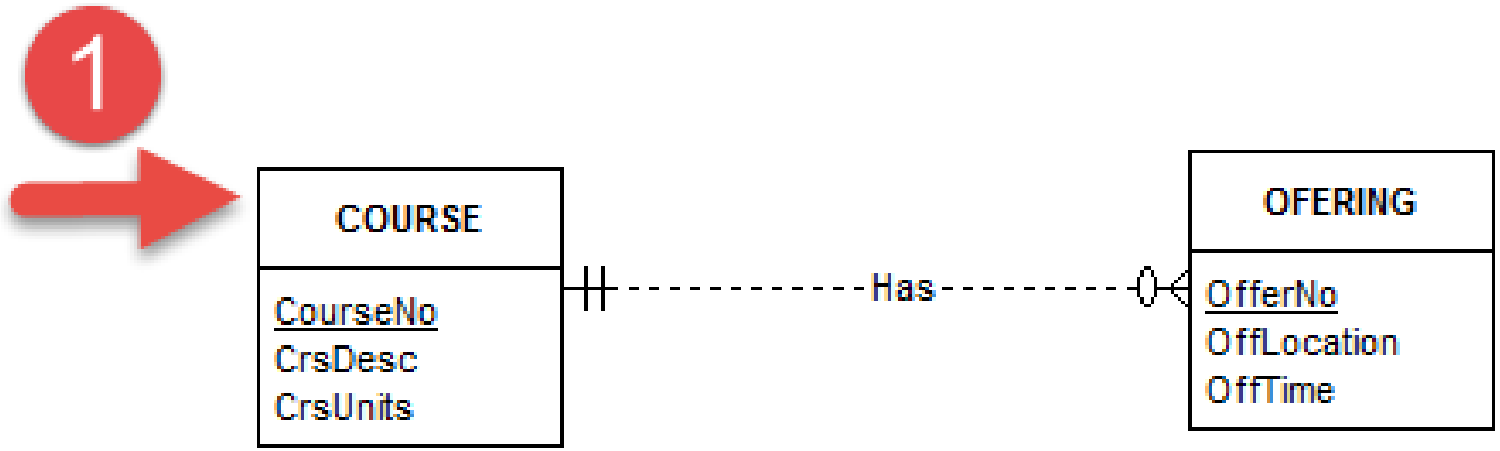
- Each entity type (except subtypes) becomes a table.
- The primary key of the entity type (if not weak) becomes the primary key of the table.
- The attributes of the entity type become columns in the table.
- This rule should be used first before the relationship rules.

Note: As you apply these rules, you can use a checklist to indicate the converted parts of an ERD.

Rule 1 Example 1

- Entity Type Rule: Each entity type converts to a table.
- Let's apply the entity type rule to the Course entity below-In step 1, the entity type rule converts the Course entity into the Course table by writing a CREATE TABLE statement.

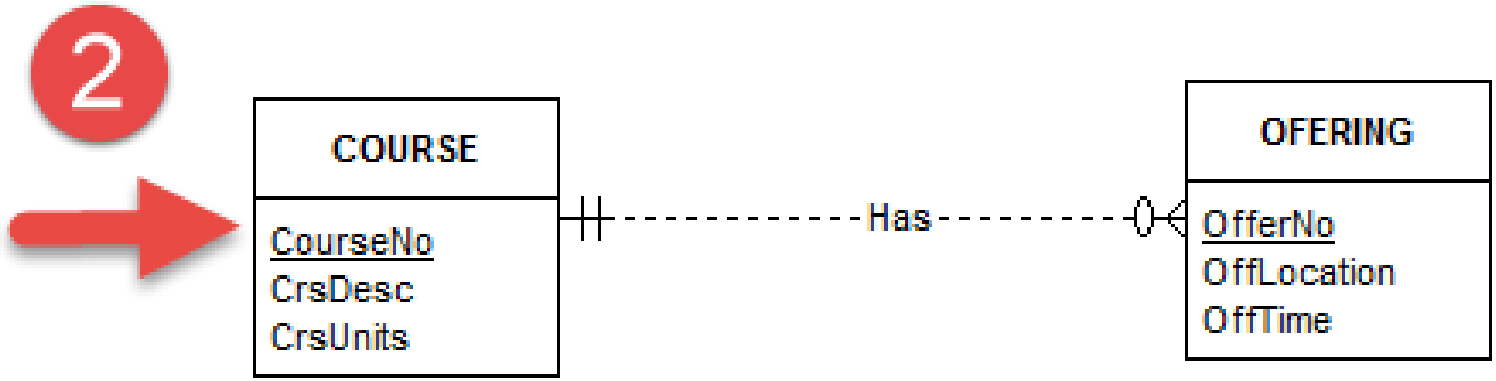
CREATE TABLE Course



Rule 1 Example 1

- The primary key of the entity type (if not weak) becomes the primary key of the table. Let's extend the CREATE TABLE statement above for the Course entity as shown in step 2 below.

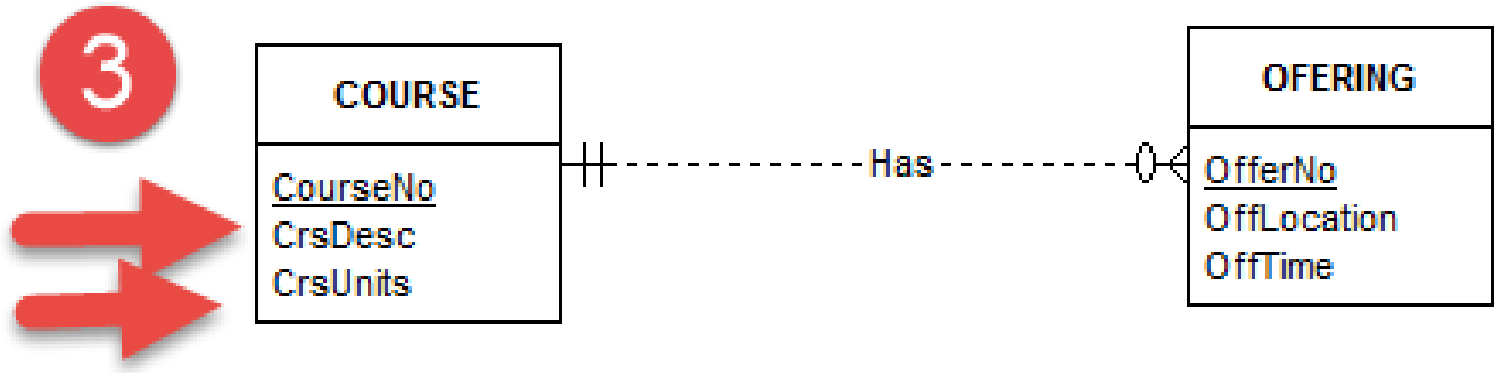
```
CREATE TABLE Course
CourseNo Char(6) Not Null,
.....
.....
Constraint CoursePK PRIMARY KEY (CourseNo)
);
```



Rule 1 Example 1

- The attributes of the entity type become columns in the table.
- This rule should be used first before the relationship rules.
- Let's extend the CREATE TABLE statement above for the Course entity as shown in step 3 below.

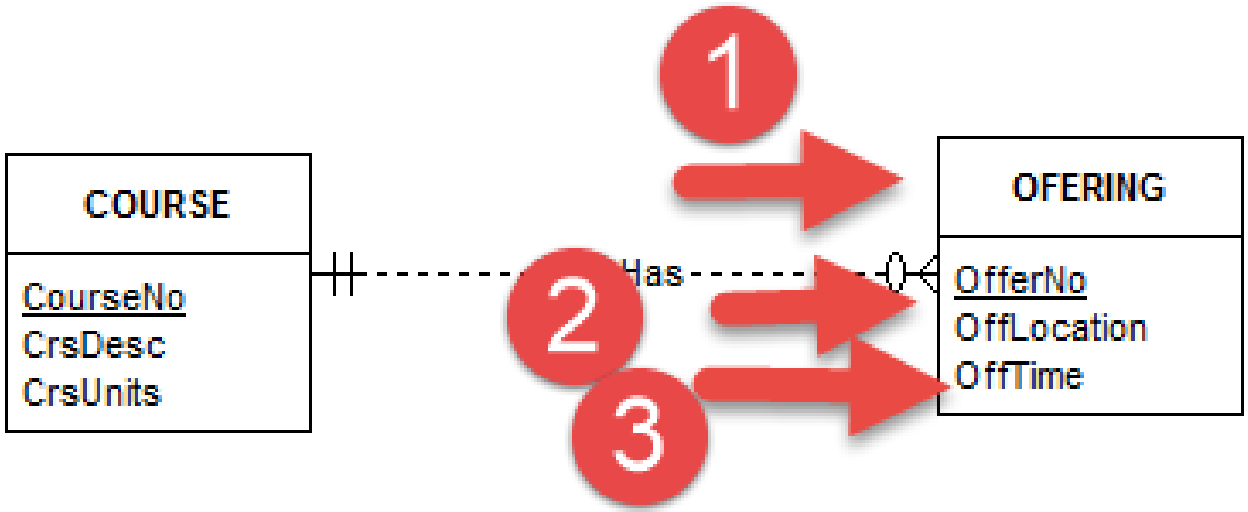
```
CREATE TABLE Course
(CourseNo char(6) not null,
crsDesc varchar2(50) not null,
CrsUnits integer,
CONSTRAINT CoursePK PRIMARY KEY (CourseNo)
);
```



Rule 1 Example 2

- Now that you have learned how to apply the entity rule in example 1 to the Course table, let's apply it to the Offering table.

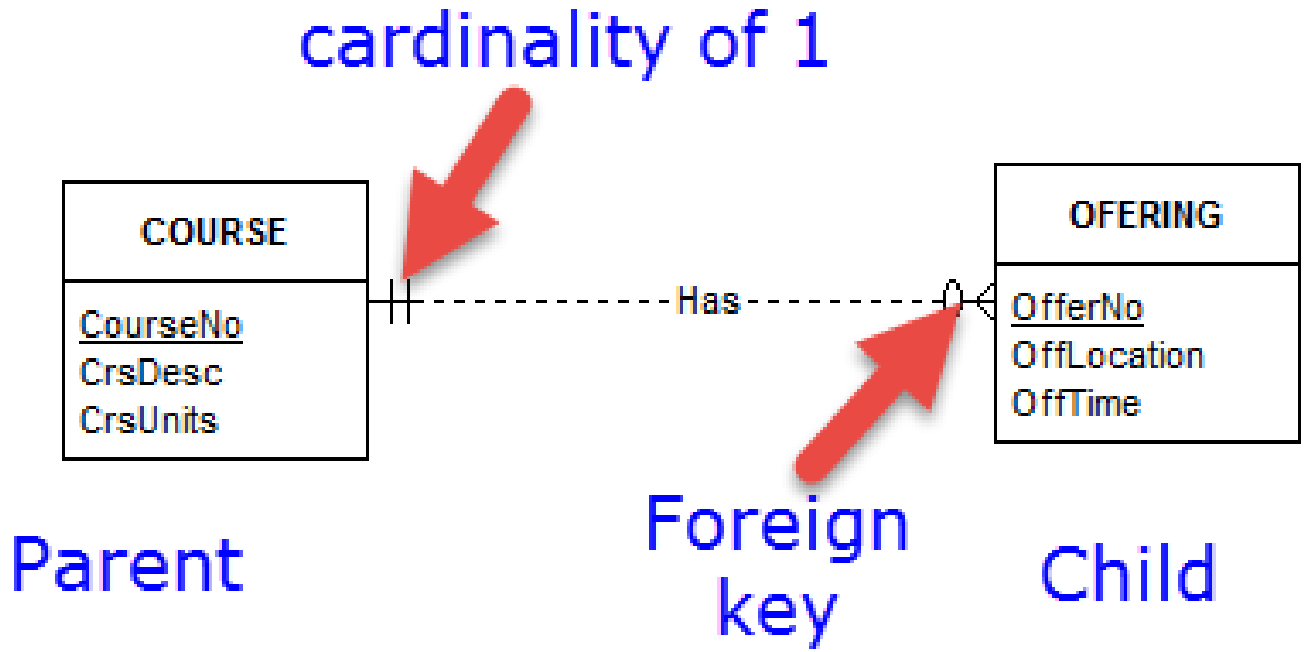
```
CREATE TABLE Offering
(OfferNo INTEGER not null,
OffLocation varchar2(30),
OffTime varchar2(10),
CONSTRAINT OfferingPK PRIMARY KEY (OfferNo)
);
```



Rule 2 – 1-M Relationship Conversion Rule

- Each 1-M relationship becomes a foreign key in the table corresponding to the child entity type (the entity type near the Crow's Foot symbol).
- If the minimum cardinality on the parent side of the relationship is one, the foreign key cannot accept null values.

Now, Let's apply the one-to-many relationship rule on the Offering table:



Rule 2 – 1-M Relationship Conversion Rule

- The one-to-many relationship rule converts HAS relationships into a foreign key.
- A foreign key should be placed into the child table.
- The child table corresponds to the entity type near the crow's foot symbol.
- In the small ERD above, the one-to-many relationship rule converts the has relationship into the CourseNo FOREIGN KEY column in the offering table.
- Note that the offering table is a child table corresponding to the entity type, nears the crow's foot symbol.

Rule 2 – 1-M Relationship Conversion Rule

- Because the has relationship is mandatory for offering entities, the CourseNo column should have a NOT NULL constraint as shown in the Offering table.

```
CREATE TABLE Offering
(OfferNo INTEGER not null,
CourseNo   not null,
OffLocation varchar2(30),
OffTime varchar2(10),
CONSTRAINT OfferingPK PRIMARY KEY (OfferNo),
CONSTRAINT CourseFK FOREIGN KEY (CourseNo) REFERENCES Course);
```

Rule 3 – M-N Relationship Conversion Rule

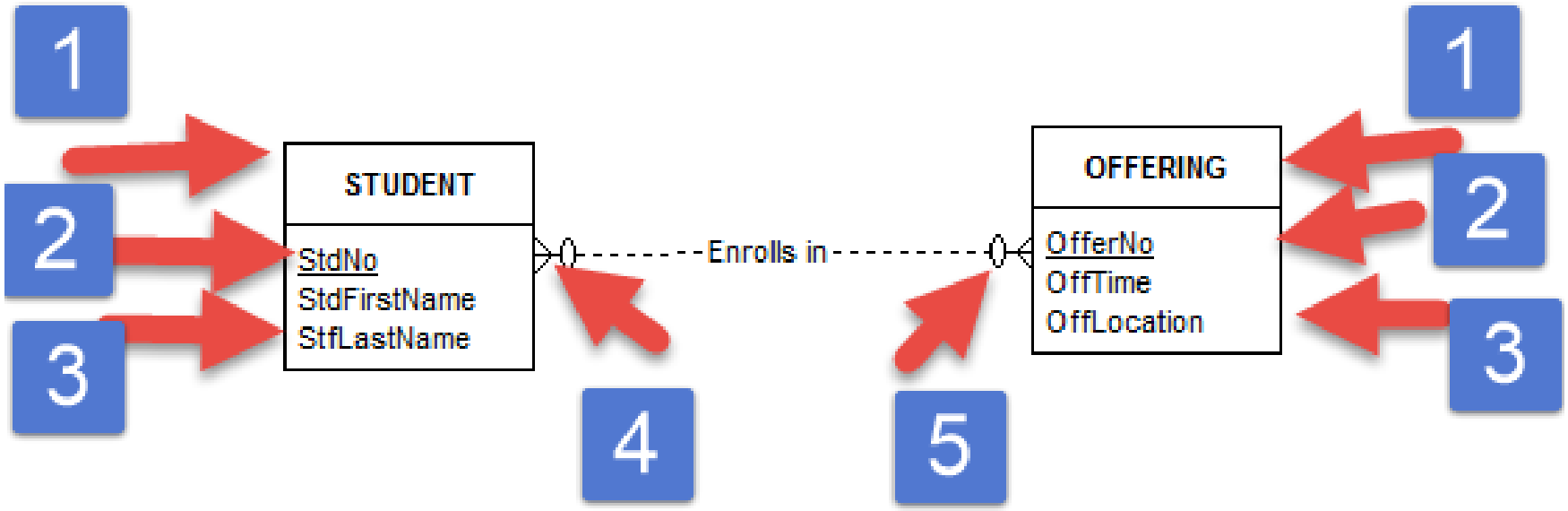
- Each M-N relationship becomes a separate table.
- The primary key of the table is a combined key consisting of the primary keys of the entity types participating in the M-N relationship.
- The many-to-many relationship rule converts each many-to-many relationship into a table with a combined primary key.
- The attributes of the many-to-many relationship become columns in a table.

Rule 3 – M-N Relationship Conversion Rule: Example

- The entity type rule converts the OFFERING and STUDENT entity types into the offering and Student tables, along with the primary key and other columns of each table.
- The many-to-many relationship rule converts the enrolls in relationship into the Enrollment table.
- The Enrollment table is a combined primary key consisting of student number and offer number.
- Both primary key components are also foreign keys. Note that the name change, enrolls in to enrollment, is optional.
- Typically table names are nouns, so renaming was used.
- Not null constraints are not used because a primary key constraint implies that null values are not allowed.

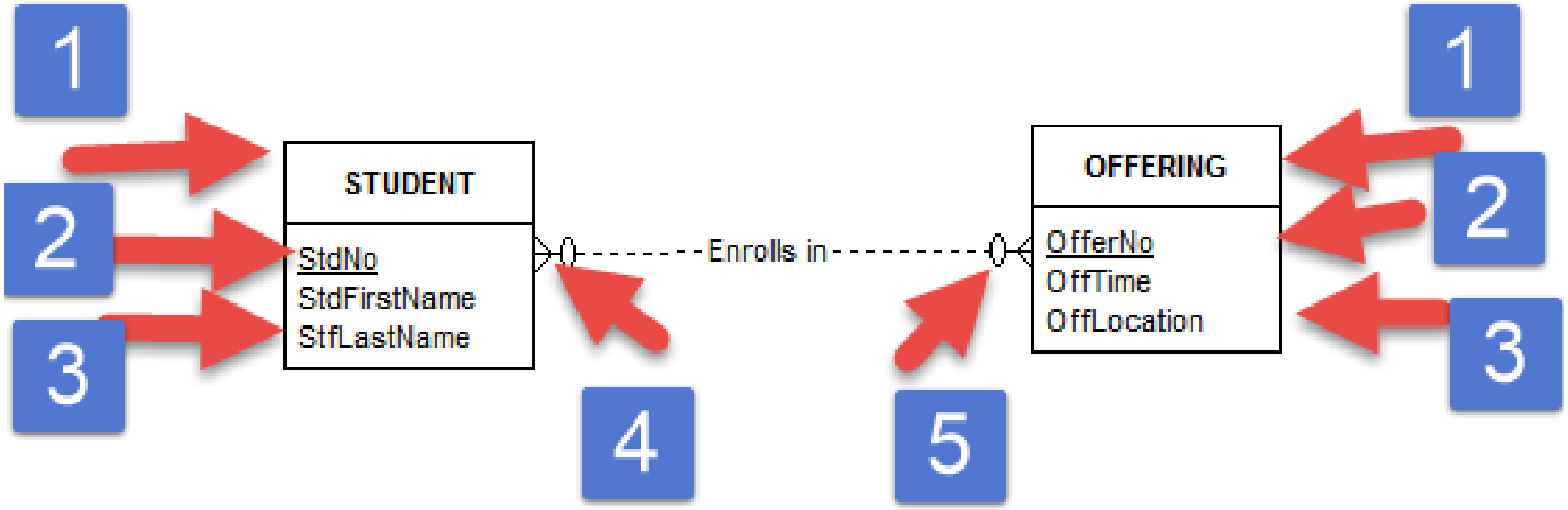
Rule 3 – M-N Relationship Conversion Rule: Example

```
CREATE TABLE Student
 (stdNo char(11) not null,
stdFirstName varchar2(30) not null,
stdLastName varchar2(30) not null,
CONSTRAINT StudentPk PRIMARY KEY (StdNo) );
```



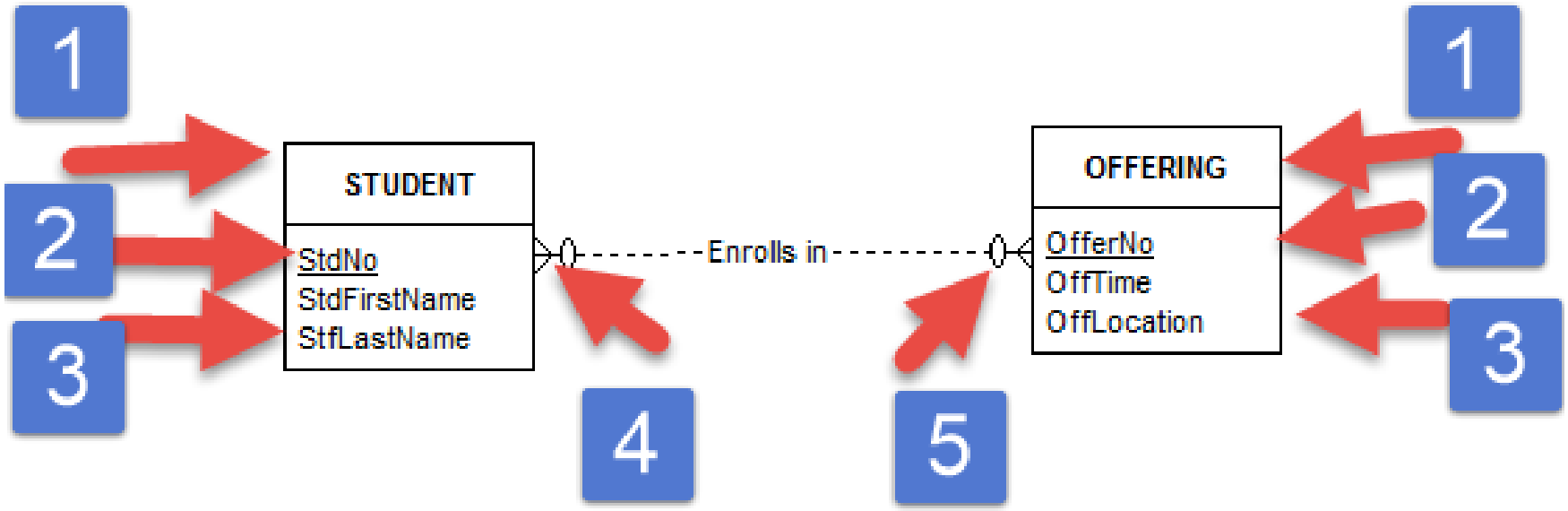
Rule 3 – M-N Relationship Conversion Rule: Example

```
CREATE TABLE Offering
(OfferNo INTEGER not null,
CourseNo   not null,
OffLocation varchar2(30),
OffTime varchar2(10),
CONSTRAINT OfferingPK PRIMARY KEY (OfferNo),
CONSTRAINT CourseFK FOREIGN KEY (CourseNo) REFERENCES Course);
```



Rule 3 – M-N Relationship Conversion Rule: Example

```
CREATE TABLE Enrollment
(OfferNo INTEGER not null,
StdNo char(11) not null,
EnrGrade decimal(3,2),
CONSTRAINT EnrollmentPK PRIMARY KEY (OfferNo, StdNo),
CONSTRAINT OfferingFK FOREIGN KEY (OfferNo) REFERENCES OfferingON DELETE CASCADE,
CONSTRAINT StudentFK FOREIGN KEY (StdNo) REFERENCES Student ON DELETE CASCADE );
```



Rule 4 - Identification Dependency Conversion Rule

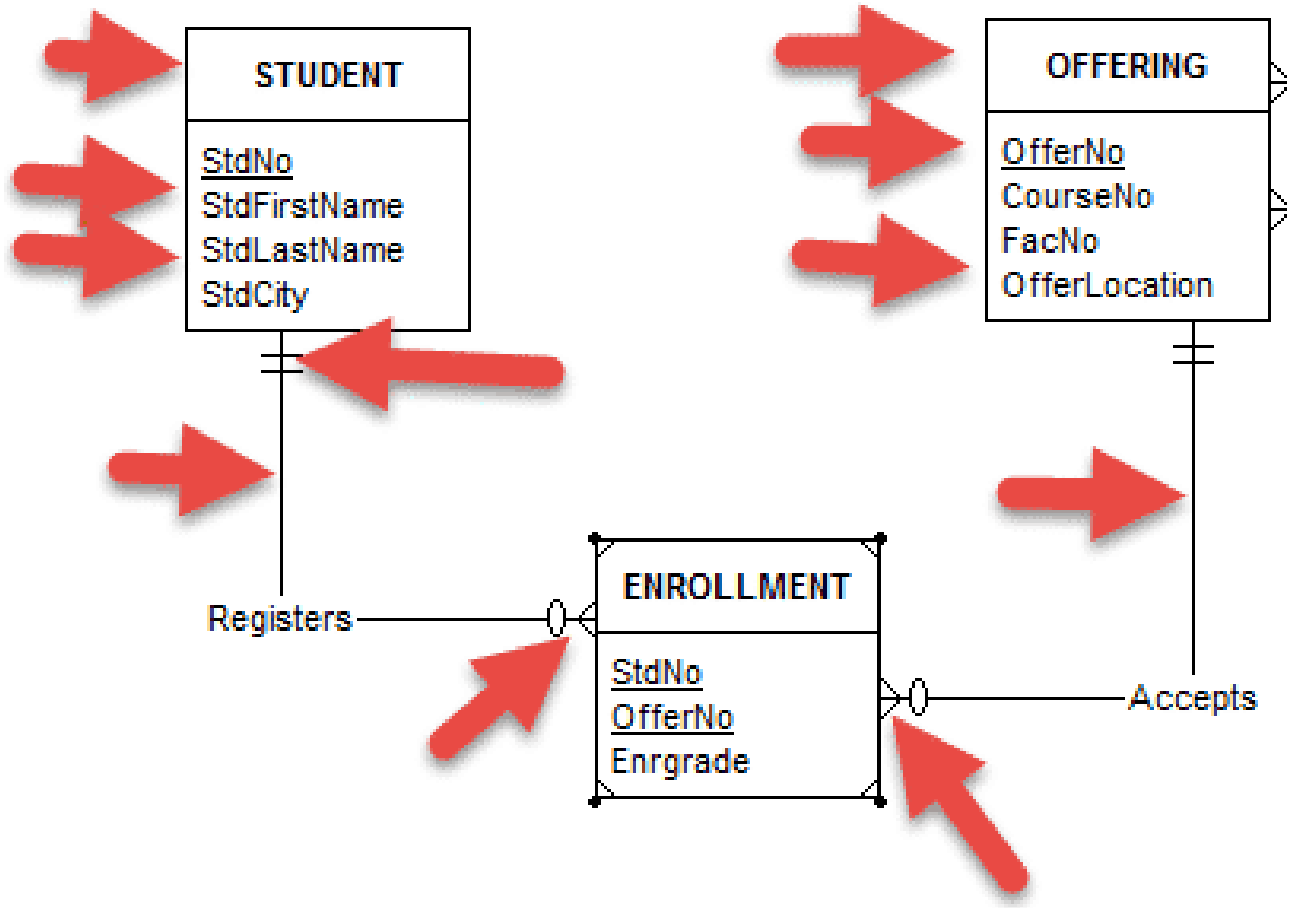
- Each identifying relationship (denoted by a solid relationship line) adds a component to a primary key.
- The primary key of the table corresponding to the weak entity consists of (i) the underlined local key (if any) in the weak entity and (ii) the primary key(s) of the entity type(s) connected by identifying relationship(s).

Example

- This small ERD consists of three tables, student, offering, and enrollment, with the same columns and constraints.
- The entity type rule is applied three times for the student, offering and enrollment tables.

Rule 4 – Identification Dependency Conversion Rule

- The one-to-many relationship rule is applied twice to create foreign keys, student number and offer number, in the enrollment table.
- The identification dependency rule is applied twice to add student number and offer number as components of the primary key in the enrollment table.



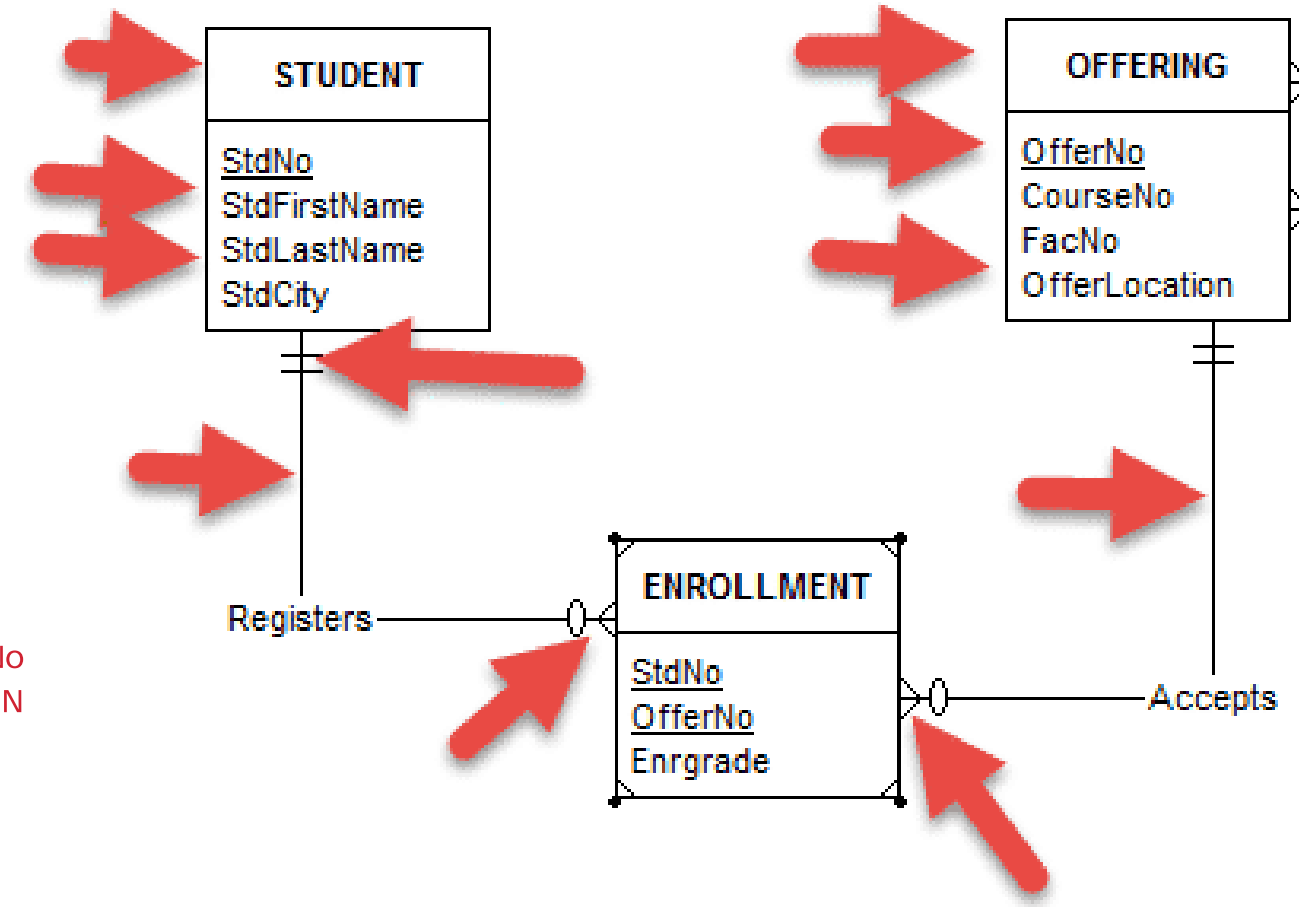
Rule 4 – Identification Dependency Conversion Rule

- The one-to-many relationship rule is applied twice to create foreign keys, student number and offer number, in the enrollment table.

The identification dependency rule is applied twice to add student number and offer number as components of the primary key in the enrollment table.Use the CASCADE option for deletions of referenced rows.

```
CREATE TABLE Student
(stdNo char(11) not null,
stdFirstName varchar2(30) not null,
stdLastName varchar2(30) not null,
CONSTRAINT StudentPk PRIMARY KEY
(StdNo) );
```

```
CREATE TABLE Offering
(OfferNo INTEGER not null,
CourseNo   not null,
OffLocation varchar2(30),
OffTime varchar2(10),
CONSTRAINT OfferingPK PRIMARY KEY (OfferNo
CONSTRAINT CourseFK FOREIGN KEY (CourseN
REFERENCES Course);
```



Rule 4 - Identification Dependency Conversion Rule

```
CREATE TABLE Enrollment
(OfferNo INTEGER not null,
StdNo char(11) not null,
EnrGrade decimal(3,2),
CONSTRAINT EnrollmentPK PRIMARY KEY
(OfferNo,
 StdNo),
CONSTRAINT OfferingFK FOREIGN KEY
(OfferNo) REFERENCES OfferingON DELETE
CASCADE,
CONSTRAINT StudentFK FOREIGN KEY (StdNo)
REFERENCES Student ON DELETE CASCADE );
```

- The most prominent difference between the entity relationship model and the relational model is the named relationships versus foreign keys.
- Two conversion rules, the one-to-many relationship rule and the many-to-many relationship rule involve this essential difference.