Database Admin

Class 2 - Modeling and SQL Language

By Ian Robert Blair, M.Sc.

Agenda

- Modeling
- Basic SQL

Normal Forms

- First Normal Form
  - Eliminate repeating groups in individual tables
  - Create a separate table for each set of related data
  - Identify each set of related data with a primary key
- Second Normal Form
  - Create separate tables for sets of values that apply to multiple records
  - Relate these tables with a foreign key
- Third Normal Form
  - Eliminate fields that do not depend on the key
- Guide to more normal forms, http://www.bkent.net/Doc/simple5.htm

Examples, pt. 1

What rules are violated by this table?

GivenNames	Surname	CourseName	Pctg
John Paul	Bloggs	Web Database Applications	72
Sarah	Doe	Programming 1	87
John Paul	Bloggs	Computing Mathematics	43
John Paul	Bloggs	Computing Mathematics	65
Sarah	Doe	Web Database Applications	65
Susan	Smith	Computing Mathematics	75
Susan	Smith	Programming 1	55
Susan	Smith	Computing Mathematics	80

Examples, pt. 2

What rules are violated by this table?

StudentID	GivenNames	Surname	CourseName	Pctg
12345678	John Paul	Bloggs	Web Database Applications	72
12345121	Sarah	Doe	Programming 1	87
12345678	John Paul	Bloggs	Computing Mathematics	43
12345678	John Paul	Bloggs	Computing Mathematics	65
12345121	Sarah	Doe	Web Database Applications	65
12345876	Susan	Smith	Computing Mathematics	75
12345876	Susan	Smith	Programming 1	55
12345303	Susan	Smith	Computing Mathematics	80

Examples, pt. 3

What rules are violated by this table?

StudentID	GivenNames	Surname	CourseName	Year	Sem	Pctg
12345678	John Paul	Bloggs	Web Database Applications	2004	2	72
12345121	Sarah	Doe	Programming 1	2006	1	87
12345678	John Paul	Bloggs	Computing Mathematics	2005	2	43
12345678	John Paul	Bloggs	Computing Mathematics	2006	1	65
12345121	Sarah	Doe	Web Database Applications	2006	1	65
12345876	Susan	Smith	Computing Mathematics	2005	1	75
12345876	Susan	Smith	Programming 1	2005	2	55
12345303	Susan	Smith	Computing Mathematics	2006	1	80

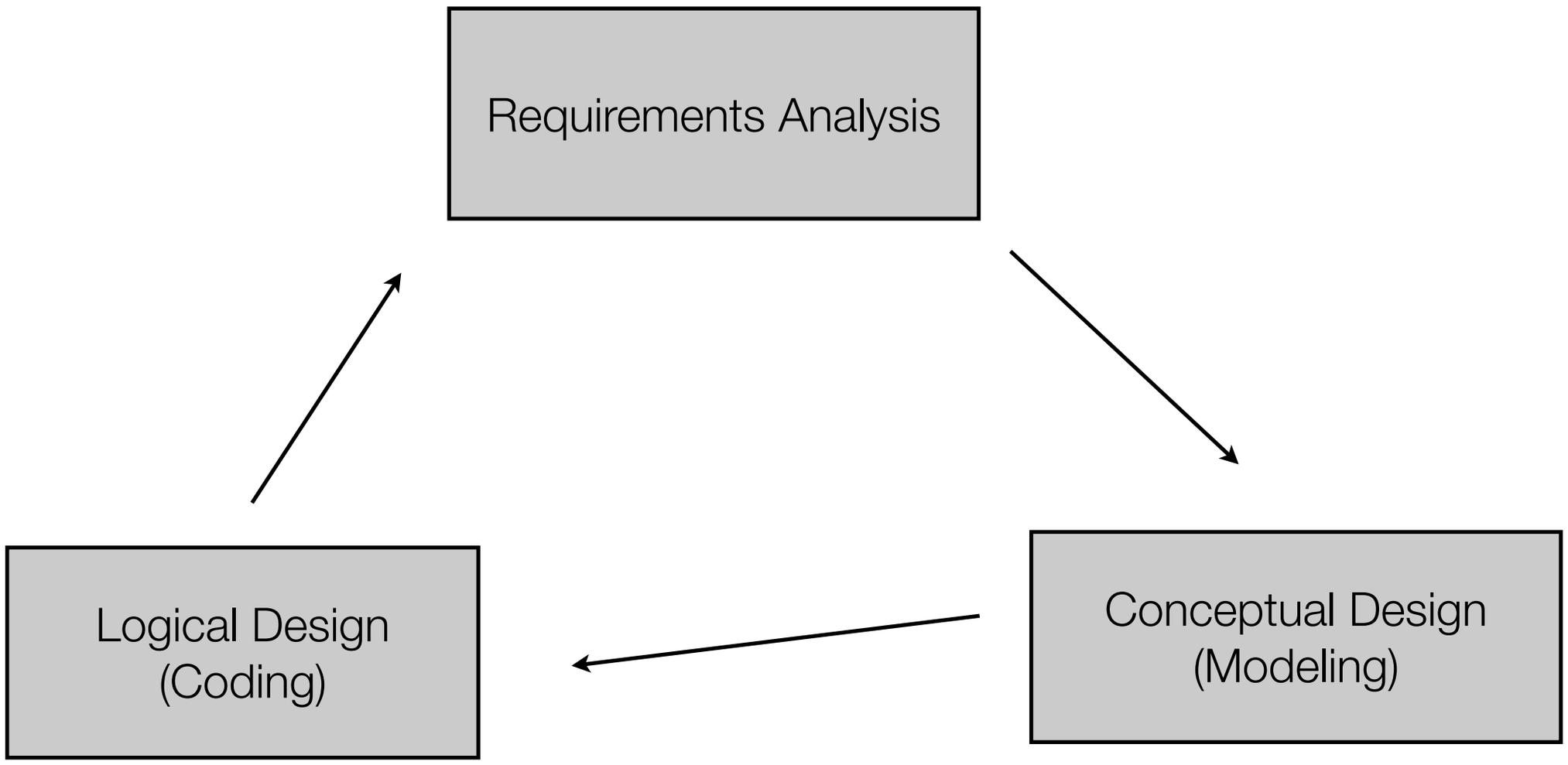
Examples, pt. 4

How can we further normalize this table?

StudentID	CourseName	Year	Sem	Pctg
12345678	Web Database Applications	2004	2	72
12345121	Programming 1	2006	1	87
12345678	Computing Mathematics	2005	2	43
12345678	Computing Mathematics	2006	1	65
12345121	Web Database Applications	2006	1	65
12345876	Computing Mathematics	2005	1	75
12345876	Programming 1	2005	2	55
12345303	Computing Mathematics	2006	1	80

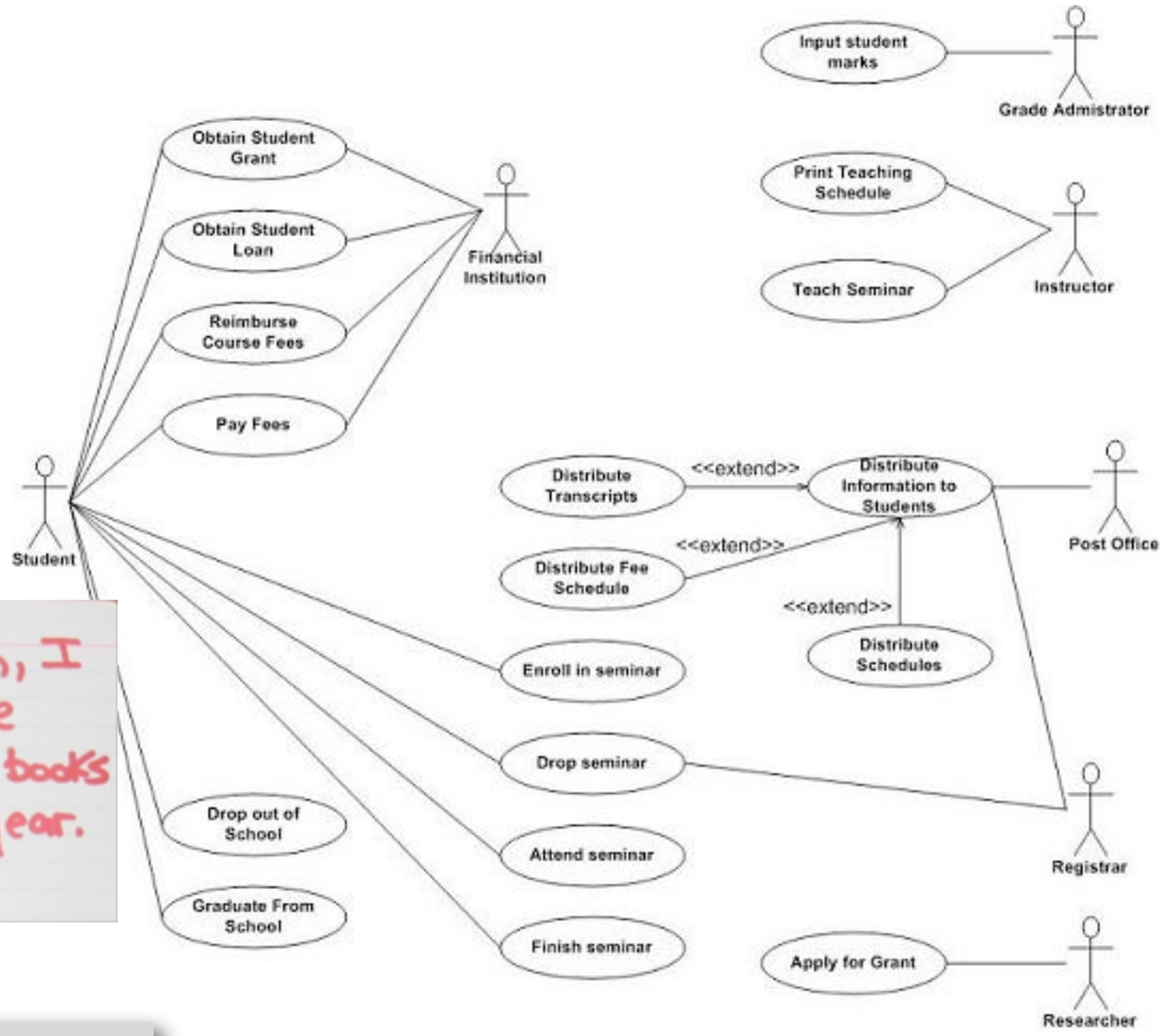
StudentID	GivenNames	Surname
12345121	Sarah	Doe
12345303	Susan	Smith
12345678	John Paul	Bloggs

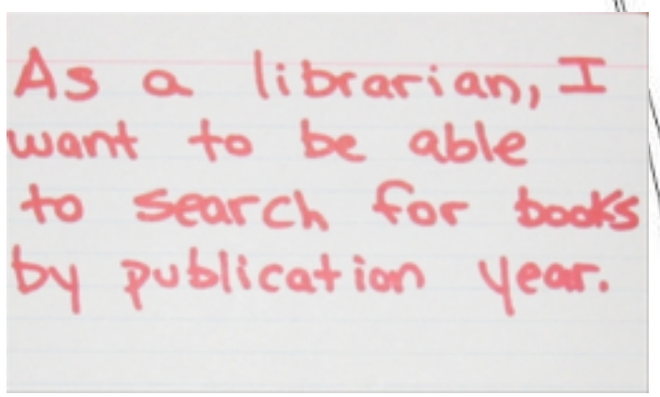
Design Process



Requirements

- User surveys or stories
- Evaluating business processes
- Developing use cases





If you want to read more about Use Cases, download: Use-Case 2.0 by Ivar Jacobson

Entity Relationship Model (ERD)

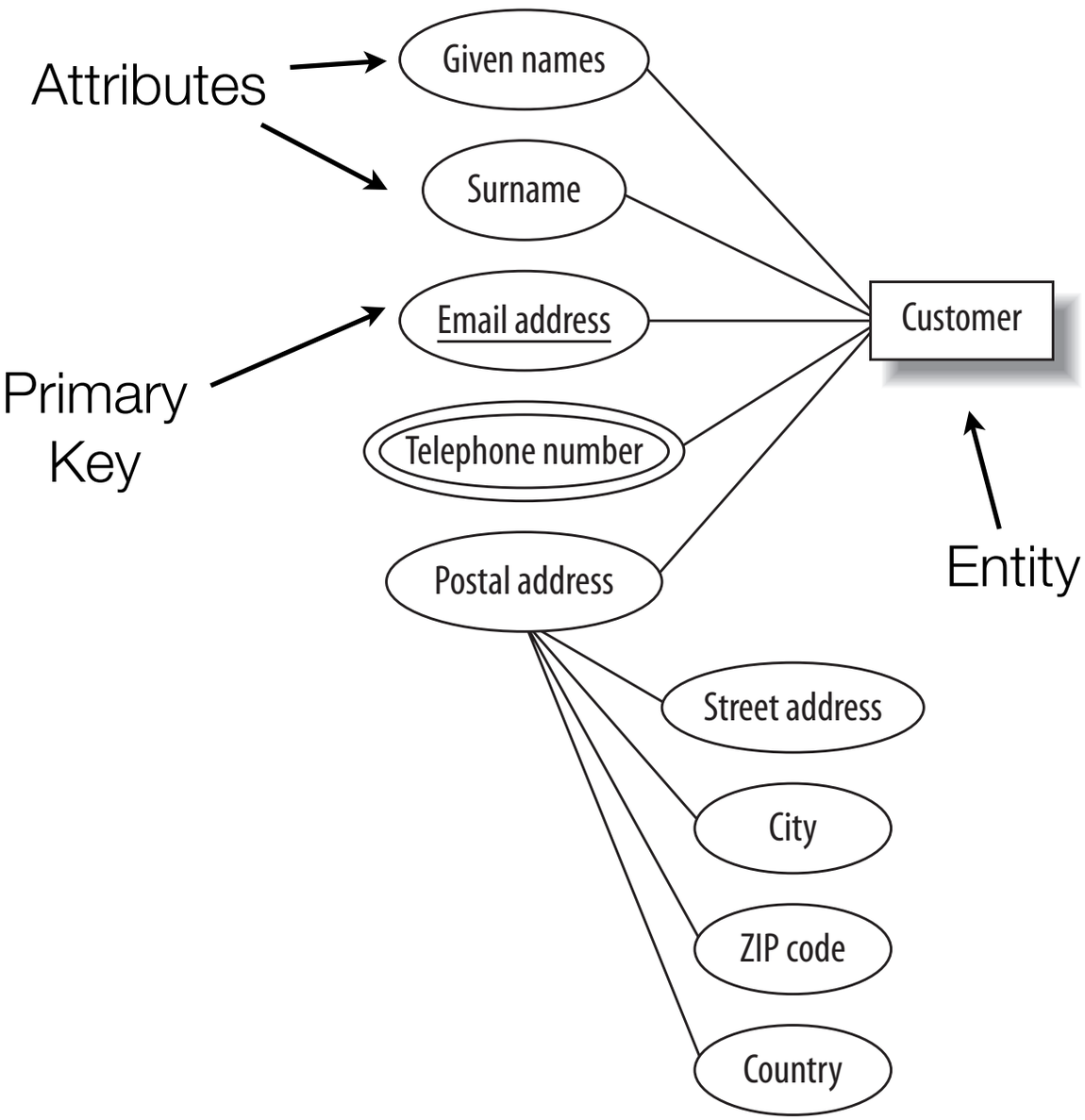
- To help visualize the design, the Entity Relationship Modeling approach involves drawing an Entity Relationship (ER) diagram

- The entity is a person, object, place or event for which data is collected

- A attribute is a characteristic of the entity (can be multivalued)

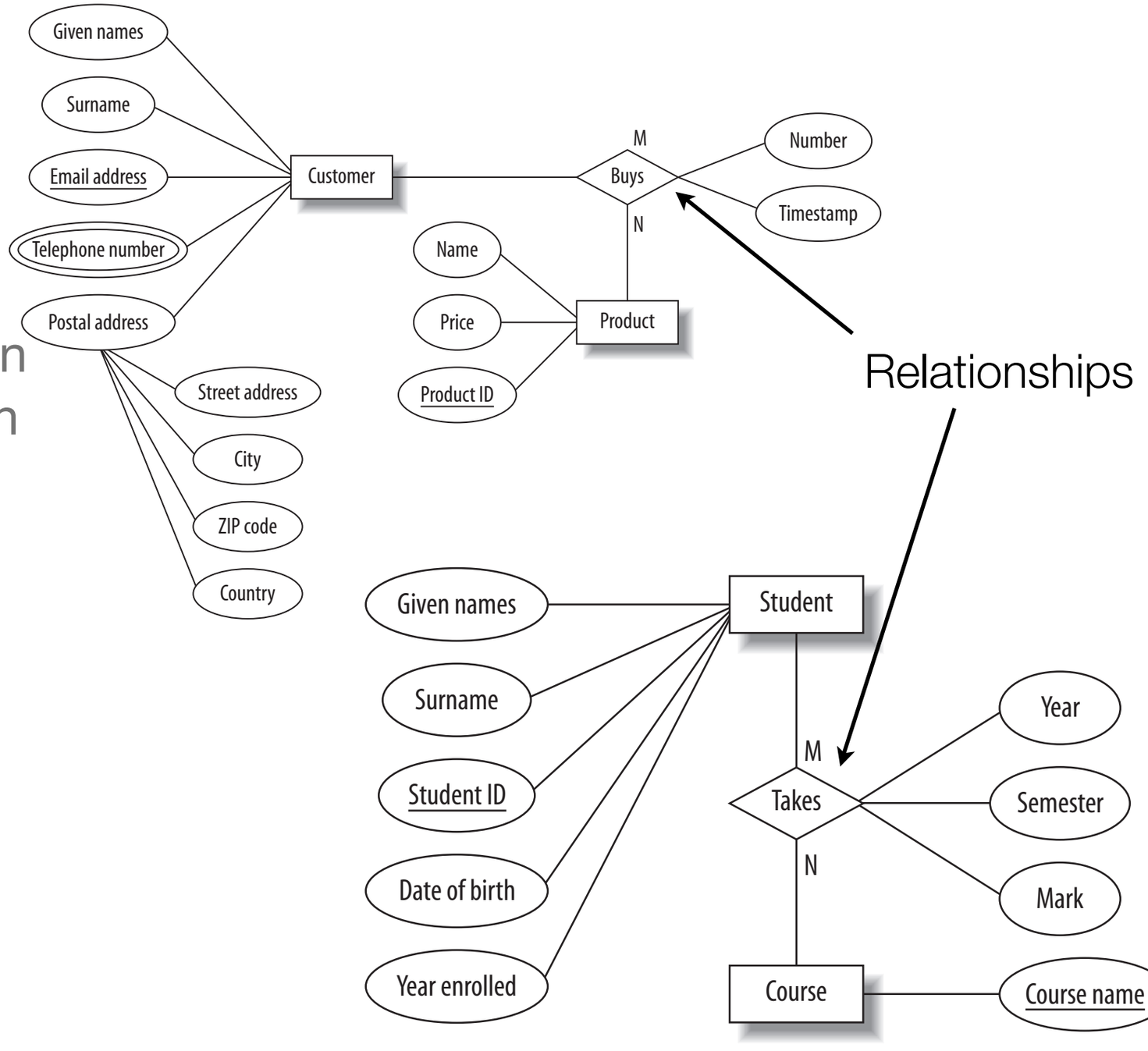
- A primary key uniquely identifies an entity

- Try to keep the design simple



Relationships

- Entities can participate in relationships with other entities
- For example, a customer can buy a product, a student can take a course, an artist can record an album, and so on
- We often use the shorthand terms 1:1, 1:N, and M:N for one-to-one, one-to-many, and many-to-many relationships, respectively

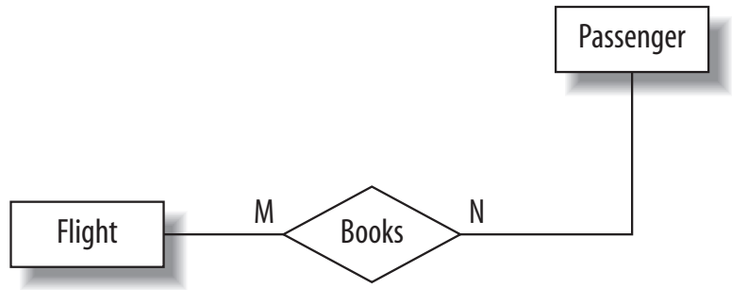


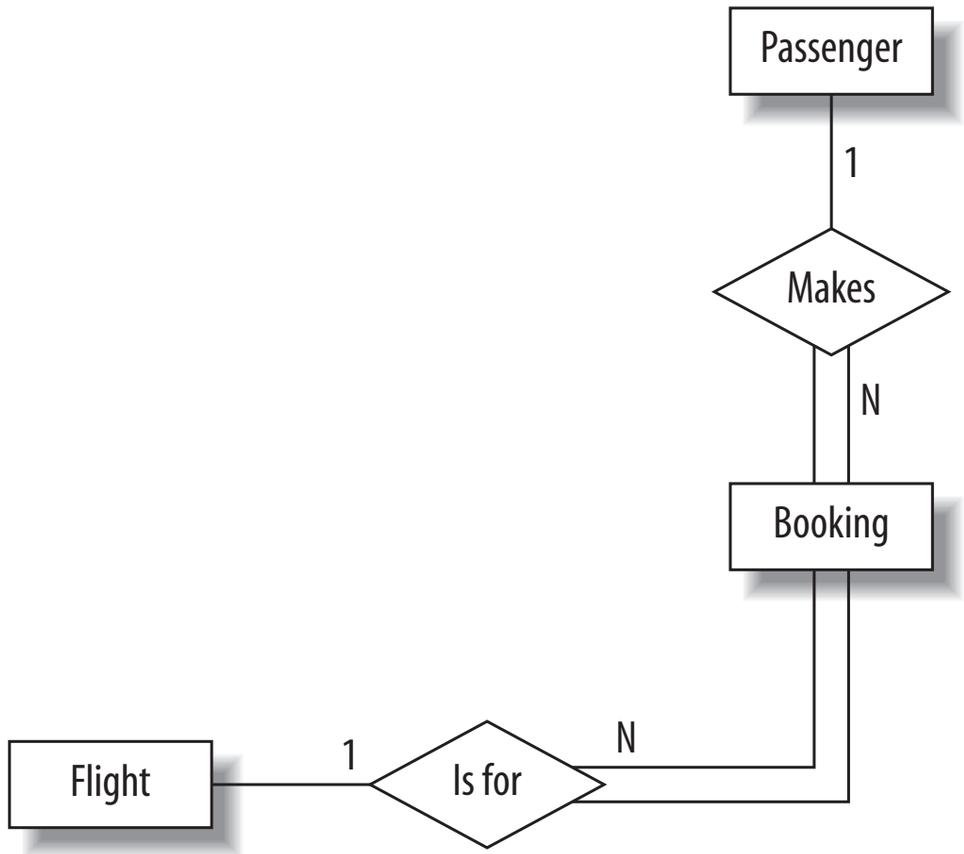
Intermediate Entities

- It is often possible to simplify many-to-many relationships by replacing the many-to-many relationship with a new intermediate entity and connecting the original entities through a many-to-one and a one- to-many relationship

- For example: “Passengers can book a seat on a flight.”

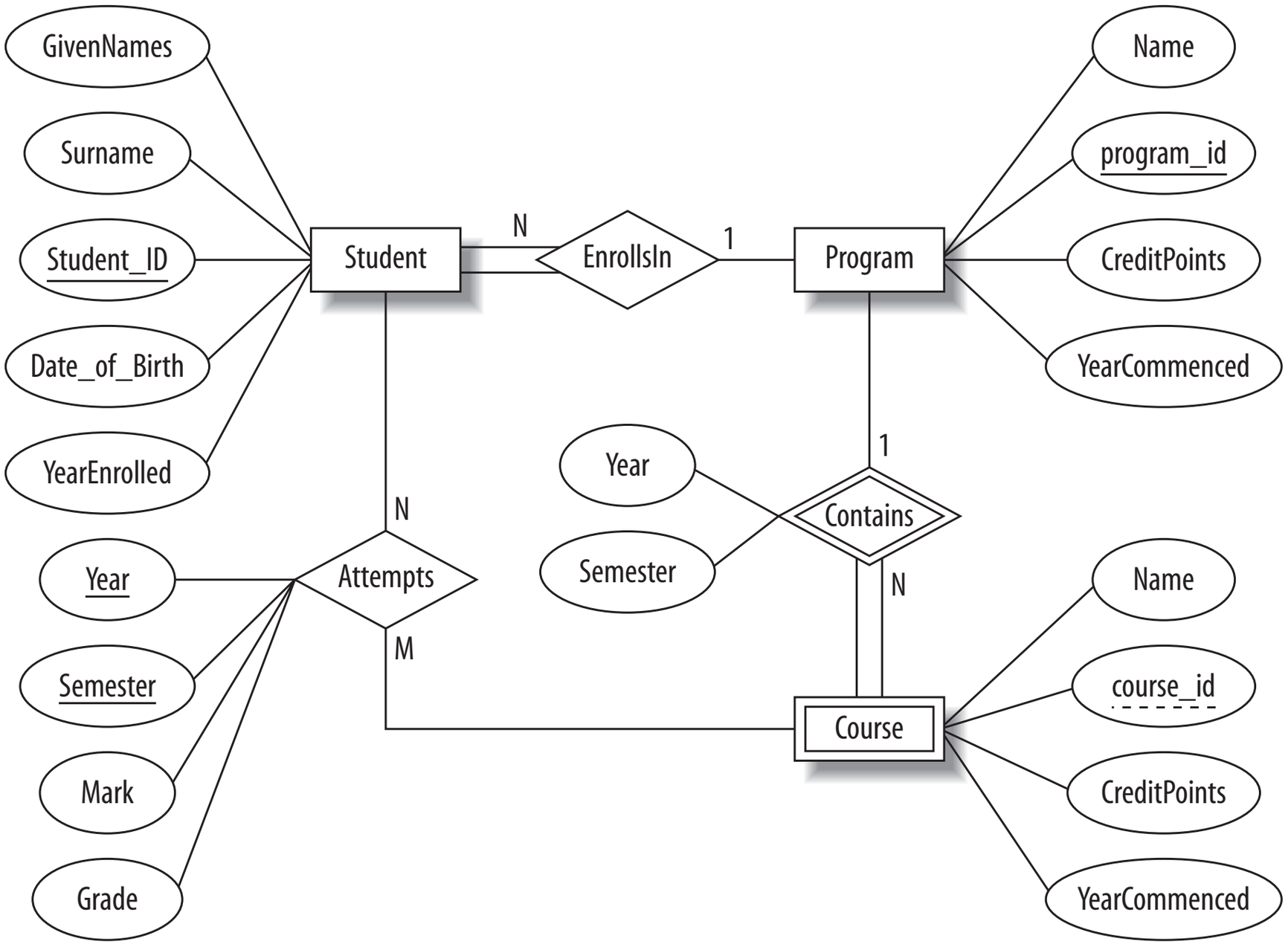
- Better: “A passenger can make a booking for a seat on a flight.”



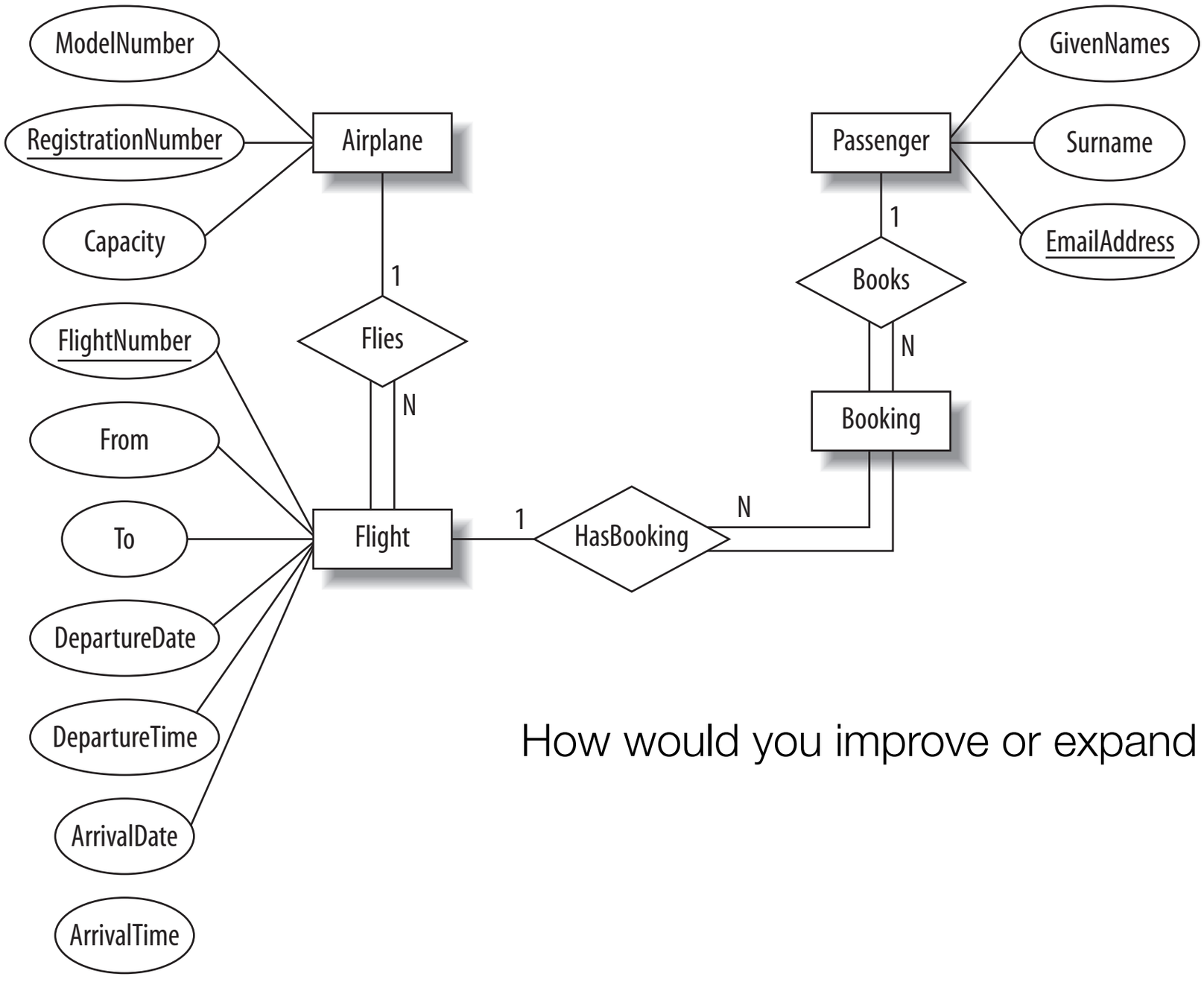


Examples: University Database

How would you improve or expand on this design?



Examples: Flight Database



How would you improve or expand on this design?

Model -> Tables, pt. 1

- For Strong Entities:
    - Create a table comprising its attributes and designate the primary key
- For each multi-valued attribute of an entity:
    - Create a table comprising the entity's primary key and the attribute

Model -> Tables, pt. 2

- For each weak entity:
    - Create a table comprising its attributes and including the primary key of its owning entity
    - The primary key of the owning entity is known as a foreign key here, because it’s a key not of this table, but of another table
    - The primary key of the table for the weak entity is the combination of the foreign key and the partial key of the weak entity
    - If the relationship with the owning entity has any attributes, add them to this table

Model -> Tables, pt. 3

- For one-to-one relationships
    - For each one-to-one relationship between two entities, include the primary key of one entity as a foreign key in the table belonging to the other
    - If one entity participates totally in the relationship, place the foreign key in its table
    - If both participate totally in the relationship, consider merging them into a single table

Model -> Tables, pt. 4

- For each non-identifying one-to-many relationship between two entities:
    - Include the primary key of the entity on the “1” side as a foreign key in the table for the entity on the “N” side.
    - Add any attributes of the relationship in the table alongside the foreign key
- For each many-to-many relationship between two entities:
    - Create a new table containing the primary key of each entity as the primary key, and add any attributes of the relationship

Model -> Tables, pt. 5

- For each relationship involving more than two entities:
    - Create a table with the primary keys of all the participating entities, and add any attributes of the relationship

Diagramming Tools

- Dia (Windows, Linux)
- Omnigraffle (Mac OSX)

SELECT Statement, pt. 1

- Basic SELECT all rows from a table
    - SELECT * FROM album;
- SELECT all from from one or more columns
    - SELECT artist_name,artist_id FROM music.artist;
- WHERE clause allows you to filter rows
    - SELECT * FROM artist WHERE artist_name = "New Order";

SELECT Statement, pt. 2

- The frequently used operators are equals (=), greater than (>), less than (<), less than or equal (<=), greater than or equal (>=), and not equal (<> or !=)
    - SELECT artist_name FROM artist WHERE artist_id < 5
    - SELECT artist_name FROM artist WHERE artist_name < 'M';
- The LIKE clause is used only with strings and means that a match must meet the pattern
- The percentage character (%) as a wildcard character that matches all possible strings
- The underscore character (_) matches exactly one wildcard character
- SELECT * FROM track WHERE track_name LIKE "R__ %";

Boolean Operations, pt. 1

- You can combine two or more conditions using the Boolean operators AND, OR, NOT, and XOR

    - SELECT album_name FROM album WHERE album_name > "C" AND album_name < "M";

    - SELECT album_name FROM album WHERE album_name LIKE "L%" OR album_name LIKE "S%" OR album_name LIKE "P%";

    - SELECT album_name FROM album WHERE album_name LIKE "L%" OR (album_name LIKE "S%" AND album_name LIKE "%g");

    - SELECT * FROM album WHERE NOT (album_id = 1 OR album_id = 3);

Boolean Operations, pt. 2

- An exclusive OR evaluates as true if only one—but not both—of the expressions is true
    - SELECT artist_name FROM artist WHERE artist_name LIKE "The%" XOR artist_name LIKE "%es";

ORDER BY

- The ORDER BY clause indicates that sorting is required, followed by the column that should be used as the sort key
    - SELECT time, track_name FROM track ORDER BY time, track_name;
    - SELECT artist_name FROM artist ORDER BY artist_name DESC;

LIMIT

- The LIMIT clause is a useful, nonstandard SQL tool that allows you to control which rows are output
    - SELECT track_name FROM track LIMIT 10;
    - SELECT track_name FROM track LIMIT 5,5;

JOIN

- JOIN allows you select data from more than one table:
    - SELECT artist_name, album_name FROM artist INNER JOIN album USING (artist_id);
    - SELECT album_name, track_name FROM album INNER JOIN track USING (artist_id, album_id) LIMIT 15;
    - SELECT played, track_name FROM track INNER JOIN played USING (artist_id, album_id, track_id) ORDER BY track.artist_id, track.album_id, track.track_id, played;

INSERT

- The INSERT statement is used to add new data to tables
    - INSERT INTO artist VALUES (7, "Barry Adamson");
    - INSERT INTO track VALUES (1, "Diamonds", 7, 1, 4.10), (2, "Boppin Out / Eternal Morning", 7, 1, 3.22), (3, "Splat Goes the Cat", 7, 1, 1.39), (4, "From Rusholme With Love", 7, 1, 3.59);
    - INSERT INTO album (artist_id, album_id, album_name) VALUES (7, 2, "Oedipus Schmoedipus");
    - INSERT INTO played SET artist_id = 7, album_id = 1, track_id = 1;

DELETE

- The DELETE statement is used to remove one or more rows from a database.
  - DELETE FROM played;
  - DELETE FROM played WHERE played < "2006-08-15";
  - DELETE FROM played ORDER BY played LIMIT 528;

TRUNCATE

- If you want to remove all rows in a table use truncate
    - TRUNCATE TABLE played;

UPDATE

- The UPDATE statement is used to change data.
    - UPDATE artist SET artist_name = UPPER(artist_name);

CREATE/DROP Databases

- Use the following to create a database:
    - CREATE DATABASE lucy;
    - CREATE DATABASE IF NOT EXISTS lucy;
- A new directory under the data directory is created for the new database and stores the text file db.opt that lists the database options;
- To delete a database:
    - DROP DATABASE music;

CREATE Tables

- Syntax:

```
CREATE TABLE IF NOT EXISTS artist (
    artist_id SMALLINT(5) NOT NULL DEFAULT 0,
    artist_name CHAR(128) DEFAULT NULL,
    PRIMARY KEY (artist_id)
);
```

COLLATION and CHARACTER SETS

- Character sets define what characters can be stored

- A collation defines how strings are ordered, and there are different collations for different languages

- SHOW CHARACTER SET;

- SHOW COLLATION;

Column Data Types, pt. 1

- INT[(width)]
  - range –2,147,483,648 to 2,147,483,647
  - my_number INT(4) ZEROFILL
- DECIMAL
  - unit_cost DECIMAL(4,2) default 0.00
- DATE
  - format stored as YYYY-MM-DD
  - alternate insertion formats allowed
- TIME
  - Stores a time in the format HHH:MM:SS
  - Input formats: [DD] HH:MM:SS, [DD] HH:MM, [DD] HH, or SS
  - Days can’t exceed 34

Column Data Types, pt. 2

- TIMESTAMP
  - Stored as YYYY-MM-DD HH:MM:SS
  - DEFAULT CURRENT_TIMESTAMP will add current time on insert
- YEAR[(digits)]
  - 2 or 4
- CHAR[(width)]
  - Stores a fixed-length string
  - maximum value of width is 255, default is 1
- VARCHAR(width)
  - Stores variable-length strings
  - maximum value of width is 65,535 characters
- BOOLEAN
  - false (zero) or true (nonzero)

Column Data Types, pt. 3

- FLOAT[(width, decimals)] or FLOAT[(precision)]
- DOUBLE[(width, decimals)]
- TEXT
  - Stores a variable amount of data (such as a document or other text file) up to 65,535 bytes in length
- ENUM('value1'[,'value2'[, ...]]
  - A list, or enumeration of string values
  - CREATE TABLE fruits_enum (fruit_name ENUM('Apple', 'Orange', 'Pear'));
- SET('value1'[,'value2'[, ...]])
  - A column of type SET can be set to zero or more values from the list value1, value2, and so on, up to a maximum of 64 different values
  - CREATE TABLE fruits_set (fruit_name SET('Apple', 'Orange', 'Pear'));

Primary Key Index

- A primary key uniquely identifies each row in a table
- A index is added by default, and its purpose is to speed up searches that use the primary key
- You can display the indexes available on a table using the SHOW INDEX command
- ALTER TABLE staff ADD PRIMARY KEY (name);

More Indexes

- More Index Examples:

- ALTER TABLE artist ADD INDEX by_name (artist_name);

- CREATE TABLE artist (
  artist_id SMALLINT(5) NOT NULL DEFAULT 0,
  artist_name CHAR(128) DEFAULT NULL,
  PRIMARY KEY (artist_id),
  KEY artist_name (artist_name)
  );

AUTO_INCREMENT

- AUTO_INCREMENT feature allows you to create a unique identifier for a row without running a SELECT query
- NOT NULL is required for AUTO_INCREMENT columns
- The column it is used on must be indexed
- The column that is it used on cannot have a DEFAULT value
- There can be only one AUTO_INCREMENT column per table

Altering Tables

- You can use the ALTER TABLE statement to add new columns to a table, remove existing columns, and change column names, types, and lengths

    - ALTER TABLE played CHANGE played last_played TIMESTAMP;

    - ALTER TABLE artist MODIFY artist_name CHAR(64) DEFAULT "Unknown";

    - ALTER TABLE artist ADD formed YEAR;

    - ALTER TABLE artist DROP formed;

    - ALTER TABLE played RENAME TO playlist

Deleting Structures

- DROP DATABASE IF EXISTS music;
- DROP TABLE IF EXISTS temp;
- ALTER TABLE artist DROP INDEX by_name;
- ALTER TABLE artist DROP PRIMARY KEY;

Aliases, pt. 1

- Allow columns to be renamed
- SELECT artist_name AS artists FROM artist;
- SELECT CONCAT(artist_name, " recorded ", album_name) AS recording FROM artist INNER JOIN album USING (artist_id) ORDER BY recording;

```
+---------------------------+
| artists                   |
+---------------------------+
| New Order                 |
| Nick Cave & The Bad Seeds |
| Miles Davis               |
| The Rolling Stones        |
| The Stone Roses           |
| Kylie Minogue             |
+---------------------------+
```

```
+-----------------------------------------------------------+
| recording                                                 |
+-----------------------------------------------------------+
| Kylie Minogue recorded Light Years                        |
| Miles Davis recorded In A Silent Way                      |
| Miles Davis recorded Live Around The World                |
| New Order recorded Brotherhood                            |
| New Order recorded Power, Corruption & Lies               |
| New Order recorded Retro - John McCready FAN              |
| New Order recorded Retro - Miranda Sawyer POP             |
| New Order recorded Retro - New Order / Bobby Gillespie LIVE |
| New Order recorded Substance (Disc 2)                     |
| New Order recorded Substance 1987 (Disc 1)                |
| Nick Cave & The Bad Seeds recorded Let Love In            |
| The Rolling Stones recorded Exile On Main Street          |
| The Stone Roses recorded Second Coming                    |
```

Aliases, pt. 2

- Aliases can also be used for tables

- SELECT ar.artist_id, al.album_name, ar.artist_name FROM album AS al INNER JOIN artist AS ar USING (artist_id) WHERE al.album_name = "Brotherhood";

```
+-----------+-------------+-------------+
| artist_id | album_name  | artist_name |
+-----------+-------------+-------------+
|         1 | Brotherhood | New Order   |
+-----------+-------------+-------------+
```

Distinct Clause

- Removes duplicates

- Query 1: SELECT DISTINCT artist_name FROM artist INNER JOIN album USING (artist_id);

- Query 2: SELECT artist_name FROM artist INNER JOIN album USING (artist_id);

```
+----------------------------+
| artist_name                |
+----------------------------+
| New Order                  |
| Nick Cave & The Bad Seeds  |
| Miles Davis                |
| The Rolling Stones         |
| The Stone Roses            |
| Kylie Minogue              |
+----------------------------+
```

```
+----------------------------+
| artist_name                |
+----------------------------+
| New Order                  |
| New Order                  |
| New Order                  |
| New Order                  |
| New Order                  |
| New Order                  |
| New Order                  |
| Nick Cave & The Bad Seeds  |
| Miles Davis                |
| Miles Davis                |
| The Rolling Stones         |
| The Stone Roses            |
| Kylie Minogue              |
```

Group By

- The GROUP BY clause sorts data into groups for the purpose of aggregation
- SELECT artist_name, COUNT(artist_name) FROM artist INNER JOIN album USING (artist_id) GROUP BY artist_name;
- SELECT artist_name, album_name, COUNT(*) FROM artist INNER JOIN album USING (artist_id) INNER JOIN track USING (artist_id, album_id) GROUP BY artist.artist_id, album.album_id;

```
+--------------------------+--------------------+
| artist_name              | COUNT(artist_name) |
+--------------------------+--------------------+
| Kylie Minogue            |                  1 |
| Miles Davis              |                  2 |
| New Order                |                  7 |
| Nick Cave & The Bad Seeds |                 1 |
| The Rolling Stones       |                  1 |
| The Stone Roses          |                  1 |
+--------------------------+--------------------+
```

```
+--------------------------+-----------------------------------+----------+
| artist_name              | album_name                        | COUNT(*) |
+--------------------------+-----------------------------------+----------+
| New Order                | Retro - John McCready FAN         |       15 |
| New Order                | Substance (Disc 2)                |       12 |
| New Order                | Retro - Miranda Sawyer POP        |       14 |
| New Order                | Retro - New Order / Bobby Gillespie |     15 |
| New Order                | Power, Corruption & Lies          |        8 |
| New Order                | Substance 1987 (Disc 1)           |       12 |
| New Order                | Brotherhood                       |       10 |
| Nick Cave & The Bad Seeds | Let Love In                      |       10 |
| Miles Davis              | Live Around The World             |       11 |
| Miles Davis              | In A Silent Way                   |        2 |
| The Rolling Stones       | Exile On Main Street              |       18 |
| The Stone Roses          | Second Coming                     |       13 |
| Kylie Minogue            | Light Years                       |       13 |
```

Aggregations

- AVG( )
- MAX( )
- MIN( )
- STD( ) or STDDEV( )
- SUM( )

Having Clause

- Adds additional control to the aggregation of rows in a GROUP BY operation
- Must contain an expression or column that's listed in the SELECT clause
- SELECT artist_name, album_name, COUNT(*) FROM artist INNER JOIN album USING (artist_id) INNER JOIN track USING (artist_id, album_id) GROUP BY artist.artist_id, album.album_id HAVING COUNT(*) > 10;

```
+--------------------+-----------------------------------------+----------+
| artist_name        | album_name                              | COUNT(*) |
+--------------------+-----------------------------------------+----------+
| New Order          | Retro - John McCready FAN               |       15 |
| New Order          | Substance (Disc 2)                      |       12 |
| New Order          | Retro - Miranda Sawyer POP              |       14 |
| New Order          | Retro - New Order / Bobby Gillespie LIVE |       15 |
| New Order          | Substance 1987 (Disc 1)                 |       12 |
| Miles Davis        | Live Around The World                   |       11 |
| The Rolling Stones | Exile On Main Street                    |       18 |
| The Stone Roses    | Second Coming                           |       13 |
| Kylie Minogue      | Light Years                             |       13 |
+--------------------+-----------------------------------------+----------+
```

Inner Join

- Joins two tables, and only rows that match are displayed
- SELECT artist_name, album_name FROM artist INNER JOIN album USING (artist_id);
- SELECT artist_name, album_name FROM artist, album WHERE artist.artist_id = album.artist_id;
- SELECT artist_name, album_name FROM artist INNER JOIN album ON artist.artist_id = album.artist_id;

```
+--------------------------+-----------------------------------------+
| artist_name              | album_name                              |
+--------------------------+-----------------------------------------+
| New Order                | Retro - John McCready FAN               |
| New Order                | Substance (Disc 2)                      |
| New Order                | Retro - Miranda Sawyer POP              |
| New Order                | Retro - New Order / Bobby Gillespie LIVE |
| New Order                | Power, Corruption & Lies                |
| New Order                | Substance 1987 (Disc 1)                 |
| New Order                | Brotherhood                             |
| Nick Cave & The Bad Seeds | Let Love In                             |
| Miles Davis              | Live Around The World                   |
| Miles Davis              | In A Silent Way                         |
| The Rolling Stones       | Exile On Main Street                    |
| The Stone Roses          | Second Coming                           |
| Kylie Minogue            | Light Years                             |
+--------------------------+-----------------------------------------+
```

Union

- Allows you to combine the output of more than one SELECT statement to give a consolidated result set
- The queries should output the same number of columns and the same type
- Results are distinct, unless you use union all
- (SELECT track_name FROM track INNER JOIN played USING (artist_id, album_id, track_id) ORDER BY played ASC LIMIT 5) UNION ALL (SELECT track_name FROM track INNER JOIN played USING (artist_id, album_id, track_id) ORDER BY played DESC LIMIT 5);

```
+-------------------------+
| track_name              |
+-------------------------+
| Fine Time               |
| Temptation              |
| Fine Time               |
| True Faith              |
| The Perfect Kiss        |
| New Blues               |
| Intruder                |
| In A Silent Way         |
```

Left/Right Join

- Includes all the data from either the left or right table in the join

- SELECT track_name, played FROM track LEFT JOIN played USING (artist_id, album_id, track_id) ORDER BY played DESC;

```
| Crystal                | 2006-08-14 10:47:21 |
| Regret                 | 2006-08-14 10:43:37 |
| Ceremony               | 2006-08-14 10:41:43 |
| The Perfect Kiss       | 2006-08-14 10:36:54 |
| True Faith             | 2006-08-14 10:30:25 |
| Temptation             | 2006-08-14 10:25:22 |
| Fine Time              | 2006-08-14 10:21:03 |
| Do You Love Me?        |                NULL |
| Nobody's Baby Now      |                NULL |
| Loverman               |                NULL |
| Jangling Jack          |                NULL |
| Red Right Hand         |                NULL |
| I Let Love In          |                NULL |
```

Nested Query

- Query using join: SELECT artist_name FROM artist INNER JOIN album USING (artist_id) WHERE album_name = "In A Silent Way";

- Query using nested query: SELECT artist_name FROM artist WHERE artist_id = (SELECT artist_id FROM album WHERE album_name = "In A Silent Way");

```
+-------------+
| artist_name |
+-------------+
| Miles Davis |
+-------------+
```

ANY/IN Subqueries

- SELECT producer_name FROM producer WHERE producer_name = ANY (SELECT engineer_name FROM engineer);

- SELECT producer_name FROM producer WHERE producer_name IN (SELECT engineer_name FROM engineer);

```
+---------------+
| producer_name |
+---------------+
| George Martin |
+---------------+
```

Exists and NOT Exists

- SELECT * FROM artist WHERE EXISTS (SELECT * FROM played);

```
+-----------+--------------------------+
| artist_id | artist_name              |
+-----------+--------------------------+
|         1 | New Order                |
|         2 | Nick Cave & The Bad Seeds |
|         3 | Miles Davis              |
|         4 | The Rolling Stones       |
|         5 | The Stone Roses          |
|         6 | Kylie Minogue            |
+-----------+--------------------------+
```

Variables

- Save values that are returned from queries
- SELECT @artist:=artist_name FROM artist WHERE artist_id = 1;
- SELECT @artist;

```
+-----------------------+
| @artist:=artist_name  |
+-----------------------+
| New Order             |
+-----------------------+
```

INSERT using SELECT

- Tracks inserted into shuffle table using select statement, which contains random tracts from albums and artists

    - INSERT INTO shuffle (artist_id, album_id, track_id) SELECT artist_id, album_id, track_id FROM track ORDER BY RAND() LIMIT 10;

```
+-----------+----------+----------+-------------+
| artist_id | album_id | track_id | sequence_id |
+-----------+----------+----------+-------------+
|         1 |        7 |        0 |           1 |
|         3 |        1 |        3 |           2 |
|         1 |        3 |       10 |           3 |
|         6 |        1 |        1 |           4 |
|         4 |        1 |        8 |           5 |
|         1 |        7 |        1 |           6 |
|         1 |        1 |        4 |           7 |
|         2 |        1 |        6 |           8 |
|         1 |        6 |        0 |           9 |
|         4 |        1 |        1 |          10 |
+-----------+----------+----------+-------------+
```

INSERT using SELECT

- Data inserted from another database, note the use of “.” to specify database
- INSERT INTO art.people (people_id, name) SELECT artist_id, artist_name FROM music.artist;
- SELECT * FROM ARTIST;

artist_id	artist_name
1	New Order
2	Nick Cave & The Bad Seeds
3	Miles Davis
4	The Rolling Stones
5	The Stone Roses
6	Kylie Minogue
60	Kylie Minogue
50	The Stone Roses
40	The Rolling Stones
30	Miles Davis
20	Nick Cave & The Bad Seeds
10	New Order

Loading Data from CSV

- On occasion data from comma delimited format (.csv) files may need to be imported or exported

- CSV is supported by most spread sheet applications like MS Excel, Numbers, or Open Calc

- LOAD DATA INFILE 'filename.csv' INTO TABLE details FIELDS TERMINATED BY ',';

- SELECT artist_name, album_name FROM artist, album WHERE artist.artist_id=album.artist_id INTO OUTFILE '/tmp/outputfile.csv' FIELDS TERMINATED BY ',';

Create Tables with Existing Data

- New tables can be created to replicate data to for backup or testing
- Doesn’t replicate indexes, they can be created after using an alter table statement
- CREATE TABLE artist_2 SELECT * from artist;

```
+-----------+-----------------------------+
| artist_id | artist_name                 |
+-----------+-----------------------------+
|         1 | New Order                   |
|         2 | Nick Cave and The Bad Seeds |
|         3 | Miles Dewey Davis           |
|         4 | The Rolling Stones          |
|         5 | The Stone Roses             |
|         6 | Kylie Minogue               |
|        10 | Jane's Addiction            |
+-----------+-----------------------------+
```

Replace Data

- The replace command will delete an existing row and insert a new row with data

- The same could be accomplished by using an UPDATE commands, which changes existing data

- Bulk replace example:

    - REPLACE artist (artist_id, artist_name) VALUES (2, "Nick Cave and The Bad Seeds"), (3, "Miles Dewey Davis");

Foreign Key Relationships

- Require the use of the InnoDB database type
- Allow cascading of updates and deletes
- May be slower than tables without relationships

- There are two ways to define a Foreign key:
  - REFERNCES<table>(<attribute>)
  - FOREIGN KEY (<attributes>) REFERNCES<table>(<attributes>)

Create Table w/ Foreign Keys

```
CREATE TABLE Studio(
    Name CHAR(30) PRIMARY KEY,
    Address VARCHAR(255),
    PresC# INT REFERNCES MovieEXec(cert#)
);
```

Lab 1

- Create diagram (using Visio or other program) for a database to do one of the following:
  - Media Sharing (movies, music, pics…) Site
  - Social Networking Site
  - Job Applications Site
- Create tables from your design

Lab 2

- Create the database schema for Lab 1
  - Write the “create table statements”
  - Write the queries for the pages
  - Write the insert statements for all forms

HW

- Read “High Performance MySQL”, Chapter 4
- Read “Learning MySQL”, Chapters 4-6
- Do page 132, Exercises 1-2
- Do page 176, Exercises 1-2