Chapter 10

Relational Model: Structure and Architecture

Relational Database

- Relational databases are the most common form of database.
- A relational database is a type of database that stores and provides access to data points that are related to one another.
- Relational databases are based on the *relational model*, an intuitive, straightforward way of representing data in tables.
- In a relational database, each row in the table is a *record* with a unique ID called the *key*. The columns of the table hold *attributes* of the data, and each record usually has a value for each attribute, making it easy to establish the relationships among data points.

Relational Database Continued

- On the provider side, many commercial (Oracle) and open source (PostgreSQL, MySQL, SQLite) database realizations exist.
- Database *modeling* and *design* also supports the requirement of eliminating redundant information. In the design process, the tables are determined through a process called *normalization* such that each one models separate information.
- The net result is a larger number of tables but designed so that redundancy is eliminated.
- Through the design and normalization process, the relationships that relate one table to another are also explicitly established.

Database Management System

Def: A ***database management system*** (DBMS) is software for managing databases and providing access to them.

- A DBMS responds to *imperatives* (“statements”) given by *application programs*, custom-written or general-purpose, executing on behalf of *users*.

- Imperatives are written in the language of the DBMS (e.g., SQL).

- Responses include completion codes, messages, and results of queries.

What does a DBMS Do?

- In response to requests given by application programs:
  - Creates and destroys variables
  - Takes note of integrity rules (constraints)
  - Takes note of authorizations (who is allowed do what, to what)
  - Updates variables (honoring constraints and authorizations)
  - Provides results of queries
  - and more…

Misuse Causes Confusion

- People often use the term “database” to refer to the DBMS. This is incorrect and a source of much confusion.

- The database is the *container* created and manipulated via the DBMS.

Structure

- In the relational database model, the structure is a set of inter-dependent tables, **where each table is comprised of rows and columns.**

- An individual table is also, in its mathematical context, called a *relation*. This is the origin of the model as a *relational* database.
  - A common misconception is that the name has to do with the relationships that might exist between tables.

Tables

- Tables are the main structures in a database.
- Each table represents a single, specific subject. The order of rows and columns within a table has no importance.
- Every table contains at least one column—known as the *primary key*—that uniquely identifies each of its rows.
- The subject of a table can be either an *object* or an *event*. When the subject is an object, the table represents something that is tangible, like a person, place, or thing.

Example

Customers

CustomerID	FirstName	LastName	StreetAddress	City	State	ZipCode
1010	Angel	Kennedy	667 Red River Road	Austin	TX	78710
1011	Alaina	Hallmark	Route 2, Box 203B	Woodinville	WA	98072
1012	Liz	Keyser	13920 S.E. 40th Street	Bellevue	WA	98006
1013	Rachel	Patterson	2114 Longview Lane	San Diego	CA	92199
1014	Sam	Abolrous	611 Alpine Drive	Palm Springs	CA	92263
1015	Darren	Gehring	2601 Seaview Lane	Chico	CA	95926

ROWS

COLUMNS

- The `Customers` table above is one of the most common examples of a table representing an object.

- `CustomerID` is the primary key of the `Customers` table.

Example

Patient Visit

PatientID	VisitDate	VisitTime	Physician	BloodPressure	Temperature
92001	2006-05-01	10:30	Ehrlich	120 / 80	98.8
96105	2006-05-02	11:00	Hallmark	160 / 90	99.1
96203	2006-05-02	14:00	Hallmark	110 / 75	99.3
97002	2006-05-01	13:00	Hallmark	112 / 74	97.5
98003	2006-05-02	9:30	Fournier	120 / 82	98.6
99014	2006-05-02	9:30	Fournier	120 / 80	98.8

- When the subject of a table is an event, the table represents something that occurs at a given point in time and has characteristics you wish to record.

- Above is a table representing an event—each row represents a doctor's appointment.

Columns

- A column is the smallest structure in the database, and it represents a characteristic of the subject of the table to which is belongs.

- In the relational model, columns of a table are also called ***fields***. We often use the term *field* synonymously with *field name*. (also called feature in machine learning.)

- The name of the column should identify its contents, such as `FirstName`, `LastName`, and `City`.

Breaking Up Data

- It is extremely important to break data into multiple columns correctly.
- For example, city, state, and ZIP code should always be separate columns.
- When you break these out, it becomes possible to sort or filter data by specific columns.

Rows

- A row represents a unique instance of the subject of the table.
- Rows are often called ***records***.
- Each row is identified throughout the database by a unique value in the primary key column(s) of that row.

Example

Customers

CustomerID	FirstName	LastName	StreetAddress	City	State	ZipCode
1010	Angel	Kennedy	667 Red River Road	Austin	TX	78710
1011	Alaina	Hallmark	Route 2, Box 203B	Woodinville	WA	98072
1012	Liz	Keyser	13920 S.E. 40th Street	Bellevue	WA	98006
1013	Rachel	Patterson	2114 Longview Lane	San Diego	CA	92199
1014	Sam	Abolrous	611 Alpine Drive	Palm Springs	CA	92263
1015	Darren	Gehring	2601 Seaview Lane	Chico	CA	95926

ROWS

COLUMNS

- In the `Customers` table, each row represents a unique customer within the table, and the `CustomerID` column identifies a given customer throughout the database.

Functional Dependencies

- Consider the example of a table maintaining records for each of the courses offered at a college.

- The fields might include a `courseid`, used internally by the registrar to uniquely identify the course, and also might have `coursesubject` like ECON or MATH, and a `coursenum`, like 201, or 350. Students would look in the catalog for ECON-350 for the description of the course.

- Beyond these, we assume fields/variables of `coursetitle`, the full title of the course, and `coursehours`, an integer for the number of credit hours of the course.

Functional Dependency Continued

- In this example we can write the functional dependency: (LHS must be unique)

`courseid` ⇒ `coursesubject, coursenum, coursetitle, coursehours`

- If we can argue that the combination of `coursesubject` and `coursenum` is unique, we can, with equal validity, write:

`coursesubject, coursenum` ⇒ `courseid, coursetitle, coursehours`

Functional Dependency Continued

- If `studentid` is unique, it can serve as the independent variable side of the functional dependency of the `students` table:

`studentid` ⇒ `studentlast, studentfirst,` *other dependent vars*

- It may be tempting to try to identify students by their first and last name, as shown in the following potential dependency:

`studentlast, studentfirst` ⇒ `studentid,` *other dependent vars*

- However, if the possibility exists for the combination of last name and first name to *not* be unique, then this would be *invalid*, because it is not a functional dependency.

Keys

- A ***key*** is one or more fields (columns) that can potentially be used to *uniquely* identify an individual record (row).

- In the relational model, given a value for a key, there is only one record that corresponds to the given key.

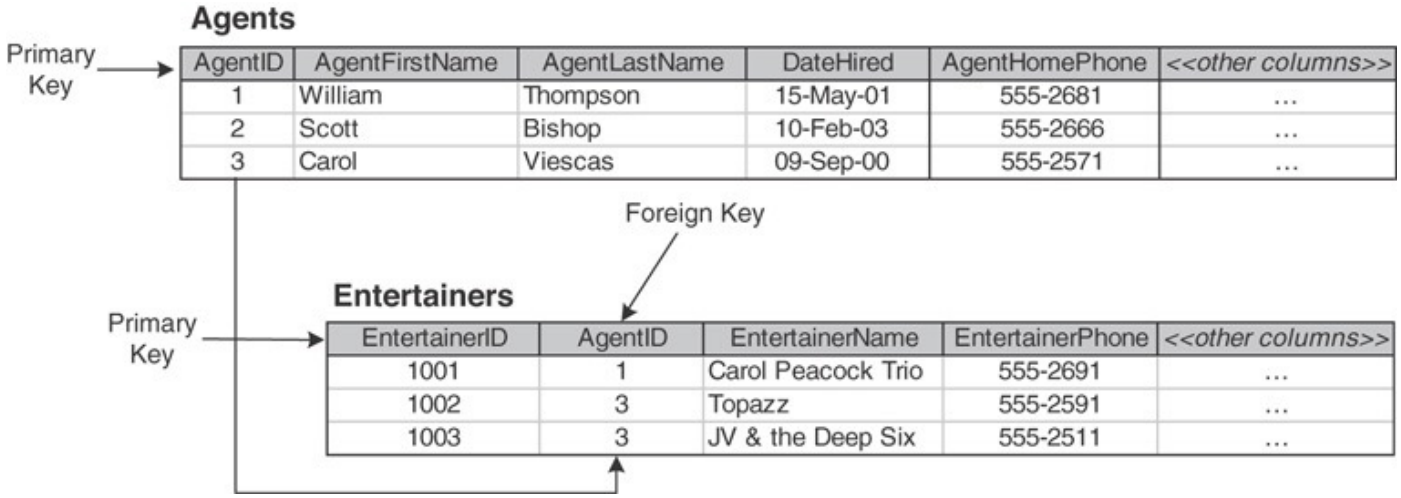
- If a key is composed of more than one field, the key is called a ***composite key***.

Primary and Foreign Keys

- The design of a relational database must identify, for every table, a ***primary key***.
  - A primary key uniquely identifies each row within a table.

- A ***foreign key*** is a column (or group of columns) in a relational database that provide a link between data in two tables.
  - The term foreign key is derived from the fact that the second table already has a primary key.

Primary and Foreign Keys



- `AgentID` is the primary key of the `Agents` table, and it is the foreign key in the `Entertainer` table.
- The `Entertainer` table already has a primary key—`EntertainerID`.
- In this relationship, `AgentID` is the column that establishes the connection between the `Agents` and `Entertainers` tables.

Views

- A ***view*** is a virtual table composed of columns from one or more tables in the database.
- The tables that comprise the view are known as ***base tables***.
- The relational model refers to a view as *virtual* **because it draws data from base tables rather than storing any data on its own.**
- Views enable you to see information in a database from may perspectives.

A Sample View

Customers

CustomerID	CustFirstName	CustLastName	CustPhone	*<<other columns>>*
10001	Doris	Hartwig	555-2671	…
10002	Deb	Waldal	555-2496	…
10003	Peter	Brehm	555-2501	…
<< more rows here >>				

Engagements

EngagementNumber	CustomerID	StartDate	EndDate	StartTime	*<<other columns>>*
3	10001	2016-09-10	2016-09-15	13:00	…
13	10003	2016-09-17	2016-09-20	20:00	…
14	10001	2016-09-24	2016-09-29	16:00	…
17	10002	2016-09-29	2016-10-02	18:00	…
<< more rows here >>					

Customer_Engagements *(view)*

EngagementNumber	CustFirstName	CustLastName	StartDate	EndDate
3	Doris	Hartwig	2016-09-10	2016-09-15
13	Peter	Brehm	2016-09-17	2016-09-20
14	Doris	Hartwig	2016-09-24	2016-09-29
17	Deb	Waldal	2016-09-29	2016-10-02
<< more rows here >>				

Relationships

- If rows of one table are associated with rows of another table, the tables are said to have a *relationship*.
- Three types of relationships can exist between tables: *one-to-one*, *one-to-many*, and *many-to-many*.
- Understanding relationships is crucial to how multi-table queries are designed and work.

One-to-One Relationship

- A pair of tables have a ***one-to-one relationship*** when a single row in the first table is related to *only one* row in the second table, and a single row in the second table is related to *only one* row in the first table.

- In this relationship, one table is referred to as the ***primary table***, and other is referred to as the ***secondary table***.
  - The primary key of the primary table is inserted into the secondary table, where it becomes a foreign key.

- Thus, the foreign key also acts as the primary key of the secondary table.

One-to-One Example

Agents

AgentID	AgentFirstName	AgentLastName	DateOfHire	AgentHomePhone	*<<other columns>>*
1	William	Thompson	1997-05-15	555-2681	…
2	Scott	Bishop	1998-02-05	555-2666	…
3	Carol	Viescas	1997-11-19	555-2571	…

Compensation

AgentID	Salary	CommissionRate	*<<other columns>>*
1	$35,000.00	4.00%	…
2	$27,000.00	4.00%	…
3	$30,000.00	5.00%	…

One-to-Many Relationship

- In a ***one-to-many relationship***, a single row in the first table can be related to *many* rows in the second table, but a single row in the second table can be related to only *one* row in the first.

- This relationship is established by taking the primary key of the table on the “one” side and inserting it into the table on the “many” side, where it becomes a foreign key.

One-to-Many Example

Entertainers

EntertainerID	EntertainerName	EntertainerPhone	*<<other columns>>*
1001	Carol Peacock Trio	555-2691	…
1002	Topazz	555-2591	…
1003	JV & the Deep Six	555-2511	…

Engagements

EngagementID	EntertainerID	CustomerID	StartDate	EndDate	*<<other columns>>*
5	1003	10006	2007-09-11	2007-09-14	…
7	1002	10004	2007-09-11	2007-09-18	…
10	1003	10005	2007-09-17	2007-09-26	…
12	1001	10014	2007-09-18	2007-09-26	…

Many-to-Many Relationship

- A pair of tables is in a ***many-to-many relationship*** when a single row in the first table can be related to *many* rows in the second table, and a single row in the second table can be related to *many* rows in the first table.

- To establish this relationship properly, you must create what is known as a ***linking table***.

- A linking table provides an easy way to associate rows from one table with those of the other and helps to ensure that you have no problems adding, deleting, or modifying any related data.

Linking Table

- A linking table is defined by taking a copy of the primary key of each table in the relationship and using them to form the structure of the new table.

- These columns serve two roles: together they form the composite primary key of the linking table, and separately they each serve as a foreign key.

- A many-to-many relationship that has not been properly established is said to be ***unresolved***.

Unresolved Example

Customers

CustomerID	CustFirstName	CustLastName	CustPhone	*<<other columns>>*
10001	Doris	Hartwig	555-2671	…
10002	Deb	Waldal	555-2496	…
10003	Peter	Brehm	555-2501	…

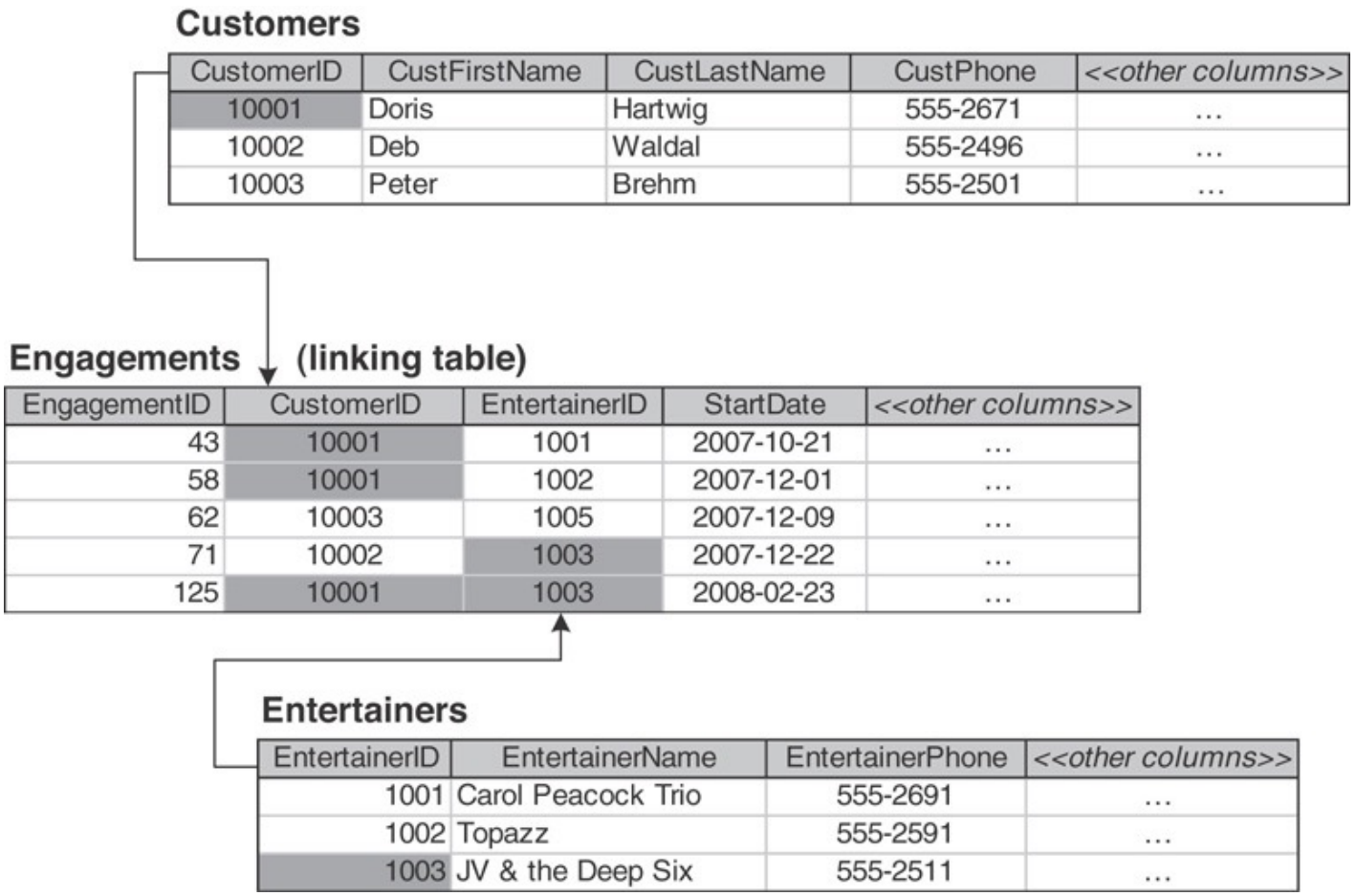
Entertainers

EntertainerID	EntertainerName	EntertainerPhone	*<<other columns>>*
1001	Carol Peacock Trio	555-2691	…
1002	Topazz	555-2591	…
1003	JV & the Deep Six	555-2511	…

- A single row in the `Customers` table can be related to many rows in the `Entertainers` tables, and vice-versa.

- How do you associate a single customer with several entertainers or a specific entertainer with several customers?

Linking Table Example



Customers

CustomerID	CustFirstName	CustLastName	CustPhone	<<other columns>>
10001	Doris	Hartwig	555-2671	…
10002	Deb	Waldal	555-2496	…
10003	Peter	Brehm	555-2501	…

Engagements **(linking table)**

EngagementID	CustomerID	EntertainerID	StartDate	<<other columns>>
43	10001	1001	2007-10-21	…
58	10001	1002	2007-12-01	…
62	10003	1005	2007-12-09	…
71	10002	1003	2007-12-22	…
125	10001	1003	2008-02-23	…

Entertainers

EntertainerID	EntertainerName	EntertainerPhone	<<other columns>>
1001	Carol Peacock Trio	555-2691	…
1002	Topazz	555-2591	…
1003	JV & the Deep Six	555-2511	…

Design Phase of a Table

- Let's consider the design of a `courses` table.

Table 10.1 Fields of the `courses` table

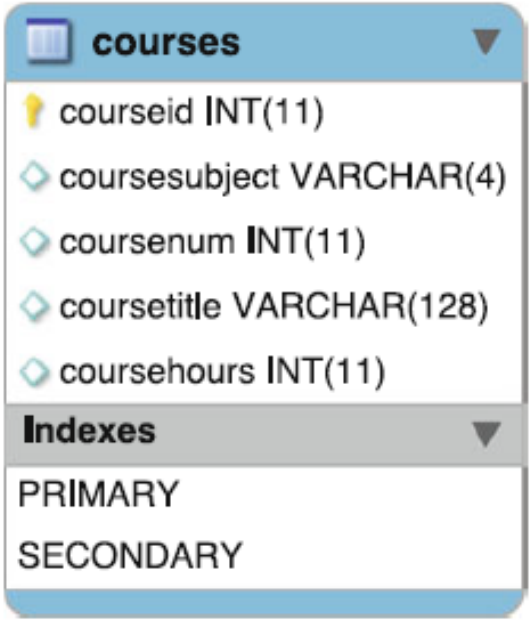
Field	Data type and size	Field description
`courseid`	Integer (4 bytes)	Unique number associated with a particular course
`coursesubject`	String of up to 4 chars	The identifier for the subject, be it department or program
`coursenum`	Integer (4 bytes)	The number of the course, as used in the catalog
`coursetitle`	String up to 128 chars	Full title of the course
`coursehours`	Integer (4 bytes)	Credit hours for course, from 1 to 4

- The dependencies are:

`courseid` → `coursesubject`, `coursenum`, `coursetitle`, `coursehours`

`coursesubject`, `coursenum` → `courseid`, `coursetitle`, `coursehours`

Table Schema Tool

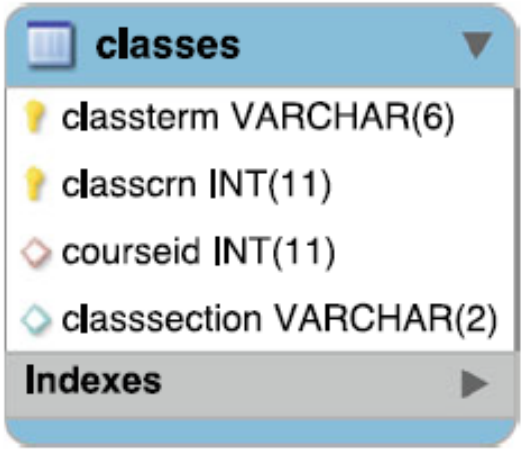


- A table is depicted in a box.
- The primary key is indicated with a key icon.
- The icon next to the other fields is either filled in, or not, indicated if a field is required to be non-NULL or not.
- The `Indexes` section refers to the auxiliary structures that allow access when given a key.
- A `PRIMARY` index will always exist. In this case, we are also showing a `SECONDARY` index that, while not shown in this picture, is the composite key of `coursesubject` and `coursenum`.

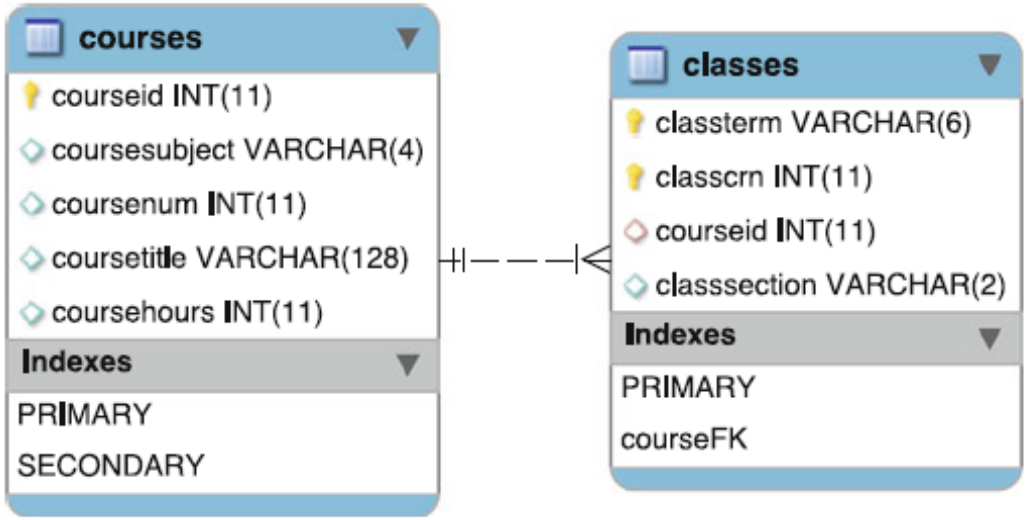
Multiple Tables Design

- The rows (records) of the `courses` table represent exactly one course, like an item in the list of courses of a catalog for a college.

- We could extend our database by adding a `classes` table, which would include a record (row) for each class instance taught.

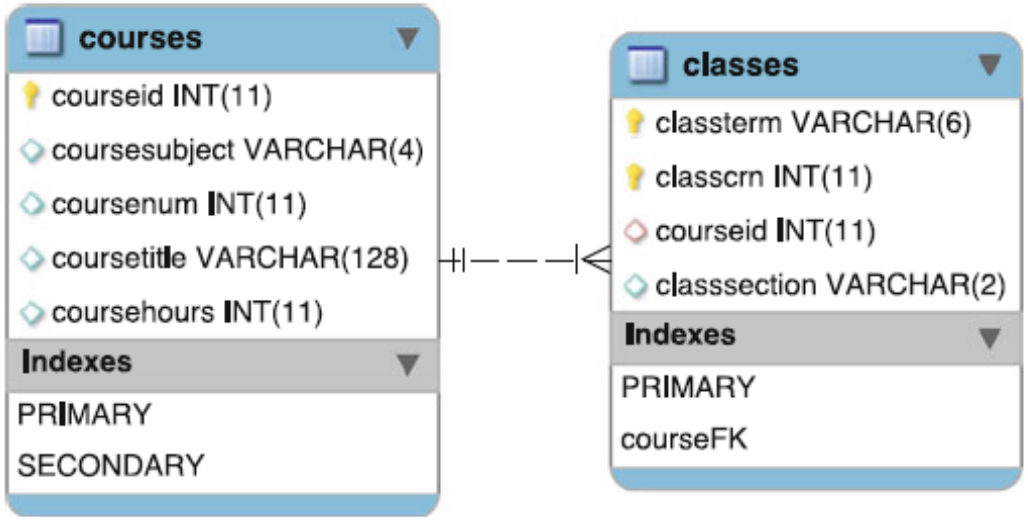


Schema for `courses` and `classes` Tables: One-to-Many Relationship



- The line denotes the relationship between the `classes` and `courses` tables.
- The “crows foot” on the `classes` side indicates that this is a *many* side of a table relationship.
- On the `courses` side, the relationship line is a hash, which indicates this is a *one* side of a table relationship.

Schema for `courses` and `classes` Tables: One-to-Many Relationship



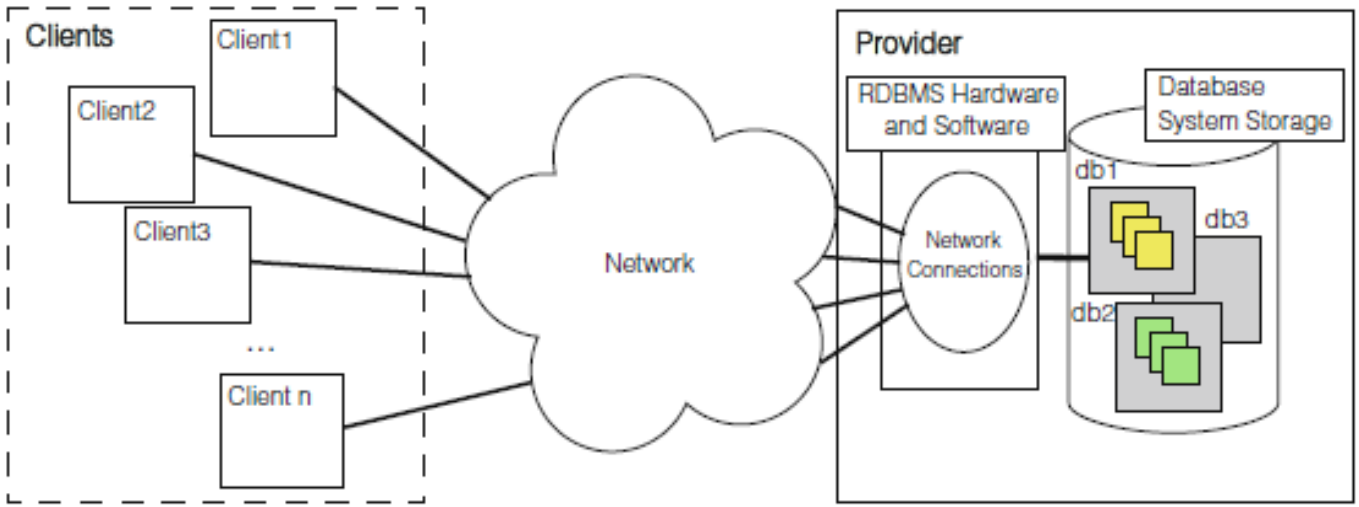
- Note the `courseFK` index for the `classes` table, which indicates that the `courseid` is a foreign key (for the `classes` table).

- Thus, we can use the `courseid` field in the `classes` table to reference the correct corresponding `courseid` in the `courses` table.

Discussion

- Note that a foreign key allows for efficient table combinations (such as a join operation).
- Relationships need to be part of the design of the database, before being populated with data.
- This can help to reduce common errors
  - If an "insert" to the `classes` table included a `courseid` that did not exist in the `courses` table, it would be rejected with an error. Such inter-table constraints are called ***referential integrity constraints***.

Database Architecture



- The sum total of the database, the DBMS, and the associated applications can be referred to as a “database system.”
- Often the term “database” is also used to loosely refer to any of the DBMS, the database system or an application associated with the database