4.1 Relational model

Database models

A ***database model*** is a conceptual framework for database systems, with three parts:

- *Data structures* that prescribe how data is organized.
- *Operations* that manipulate data structures.
- *Rules* that govern valid data.

Some database models are described in academic literature and standardized by official organizations. Others are derived informally from prominent database systems.

The ***relational model*** is a database model based on a tabular data structure. The model was published in 1970 by E. F. Codd of IBM and released in commercial products around 1980. The data structure, operations, and rules are standardized in SQL, the universal query language of relational databases.

Many non-relational database models have been published and implemented in database systems. In the 1960s and 1970s, hierarchical and network databases were dominant. At the time, computers were relatively slow and memory was limited. As a result, these databases were optimized for performance at the expense of simplicity and flexibility. Relational databases are relatively simple to manage and, as performance improved during the 1980s, rapidly displaced hierarchical and network databases.

Relational databases were initially designed for transactional data, such as bank transactions and airline reservations. The rise of the internet in the 1990s generated ***big data***, characterized by unprecedented data volumes and rapidly changing data structures. Many alternative database models and systems, optimized for big data, have appeared since 2000. However, relational databases have gradually improved support for big data and continue to dominate the commercial database market.

Table 4.1.1: Example database models.

	Primary data structure	Initial product releases	Example database system	Strengths
Hierarchical	Tree	1960s	IMS	Fast queries Efficient storage

Network	Linked list	1970s	IDMS	Fast queries Efficient storage
Relational	Table	1980s	Oracle Database	Productivity and simplicity Transactional applications
Object	Class	1990s	ObjectStore	Integration with object-oriented programming languages
Graph	Vertex and edge	2000s	Neo4j	Flexible schema Evolving business requirements
Document	XML JSON	2010s	MongoDB	Flexible schema Unstructured and semi-structured data

PARTICIPATION ACTIVITY 4.1.1: Database models.

1) Which database is relational?
   - Oracle Database
   - IDMS
   - MongoDB

2) The relational model was originally developed for which types of applications?
   - Big data storage and analysis
   - Transactional applications like banking and airline reservations
   - Desktop applications with small databases

3) What was the initial impediment to commercial adoption of relational databases in the early 1980s?

- Reliability
- Processing speed
- Cost

Relational data structure

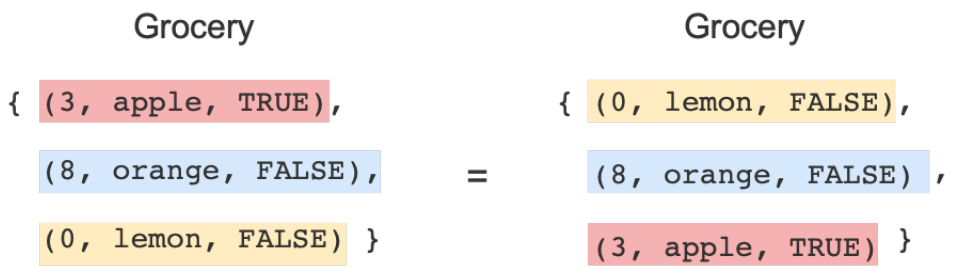
The relational data structure is based on set theory. A ***set*** is an unordered collection of elements enclosed in braces. Ex: *{a, b, c}* and *{c, b, a}* are the same, since sets are not ordered. A ***tuple*** is an ordered collection of elements enclosed in parentheses. Ex: *(a, b, c)* and *(c, b, a)* are different, since tuples are ordered.

The data structure organizes data in tables:

- A ***table*** has a name, a fixed tuple of columns, and a varying set of rows.
- A ***column*** has a name and a data type.
- A ***row*** is an unnamed tuple of values. Each value corresponds to a column and belongs to the column's data type.
- A ***data type*** is a named set of values, from which column values are drawn.

Since a table is a set of rows, the rows have no inherent order.

PARTICIPATION ACTIVITY 4.1.2: Table rows are not ordered.



Animation content:

Step 1: The Grocery table is set of three rows. A table named Grocery appears with three rows: 3, apple, TRUE

8, orange, FALSE
0, lemon, FALSE

Step 2: Since sets are not ordered , the left and right tables are the same. A second table also named Grocery appears to the right of the first table, with rows in a different order:
0, lemon, FALSE
8, orange, FALSE
3, apple, TRUE
The first row of the first Grocery table and the third row of the second Grocery table are highlighted red. The second row of the first Grocery table and the second row of the second Grocery table are highlighted blue. The third row of the first Grocery table and the first row of the second Grocery table are highlighted yellow. An equals sign appears between the two tables.

Animation captions:

1. The Grocery table is set of three rows.
2. Since sets are not ordered , the left and right tables are the same.

The terms table, column, row, and data type are commonly used in database processing. Relation, attribute, tuple, and domain are equivalent mathematical terms, often used in academic literature. File, field, record, and data type are similar terms from file processing.

Table 4.1.2: Similar data structure terms.

Databases	Mathematics	Files
Table	Relation	File
Column	Attribute	Field
Row	Tuple	Record
Data type	Domain	Data type

PARTICIPATION ACTIVITY 4.1.3: Relational data structure.

1) Which terms are commonly used in database processing?

database processing?

- Tuple, relation, attribute
- Row, table, column
- Record, file, field

2) Are these tables the same?

{ (8, mango, FALSE), (-11, watermelon, FALSE) }
{ (-11, watermelon, FALSE), (8, mango, FALSE) }

- Yes
- No

3) In the relational data structure, which components are named?

- Data type, row, table
- Data type, table
- Data type, table, column

4) Can a query select one specific row from a table?

- Yes, by specifying the row name
- Yes, by specifying one or more row values
- No

Relational operations

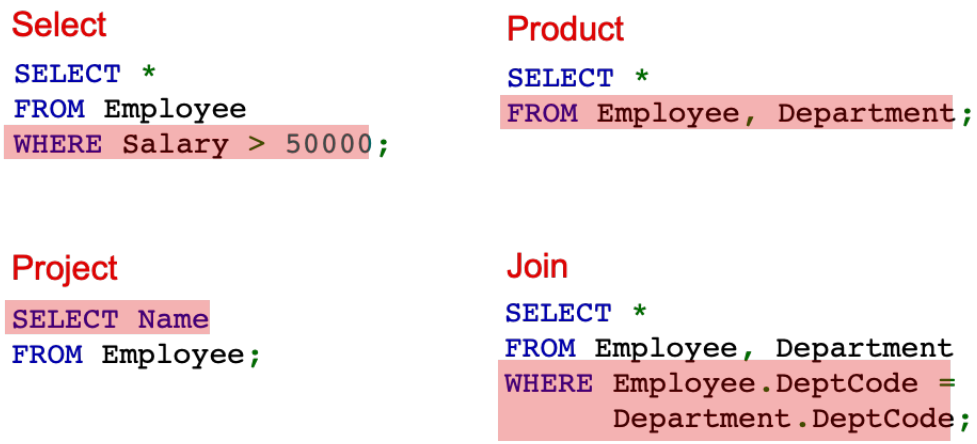
Like the relational data structure, relational operations are based on set theory. Each operation generates a result table from one or two input tables:

- *Select* selects a subset of (or all) rows of a table.
- *Project* selects one or more columns of a table.
- *Product* lists all combinations of rows of two tables.
- *Join* combines two tables by comparing related columns.
- *Union* selects all rows of two tables.

- *Union* selects all rows of two tables.
- *Intersect* selects rows common to two tables.
- *Difference* selects rows that appear in one table but not another.
- *Rename* changes a table name.
- *Aggregate* computes functions over multiple table rows, such as sum and count.

These operations are collectively called ***relational algebra*** and are the theoretical foundation of the SQL language. Since the result of relational operations is always a table, the result of an SQL query is also a table.

PARTICIPATION ACTIVITY 4.1.4: Relational operations and SQL.



Animation content:

Step 1: SELECT * selects all columns in the Employee table. SELECT * FROM Employee WHERE Salary > 50000; appears. SELECT * is highlighted.

Step 2: The select operation selects only rows for which the Salary column is > 50000. The caption Select appears. WHERE Salary > 50000 is highlighted.

Step 3: The project operation selects only the Name column. The caption Project appears. SELECT Name FROM Employee; appears. SELECT Name is highlighted.

Step 4: The product operation selects all combinations of Employee and Department rows. The

caption Product appears. SELECT * FROM Employee, Department; appears. FROM Employee, Department is highlighted.

Step 5: The join operation combines Employee and Department by comparing the tables' DepartCode columns. The caption Join appears. SELECT * FROM Employee, Department WHERE Employee.DepartmenCode = Department.DepartCode; appears. WHERE Employee.DepartmenCode = Department.DepartCode is highlighted.

Animation captions:

1. SELECT * selects all columns in the Employee table.
2. The select operation selects only rows for which the Salary column is > 50000.
3. The project operation selects only the Name column.
4. The product operation selects all combinations of Employee and Department rows.
5. The join operation combines Employee and Department by comparing the tables' DepartCode columns.

PARTICIPATION ACTIVITY 4.1.5: Relational operations.

1) What is the result of a relational operation?
   - ○ A row
   - ○ A column
   - ○ A table

2) Name three relational operations.
   - ○ Select, project, and union
   - ○ Square root, exponent, and logarithm
   - ○ Integrate and differentiate

3) An SQL statement can implement only one relational operation.
   - ○ True
   - ○ False

Relational rules

Rules are logical constraints that ensure data is valid.

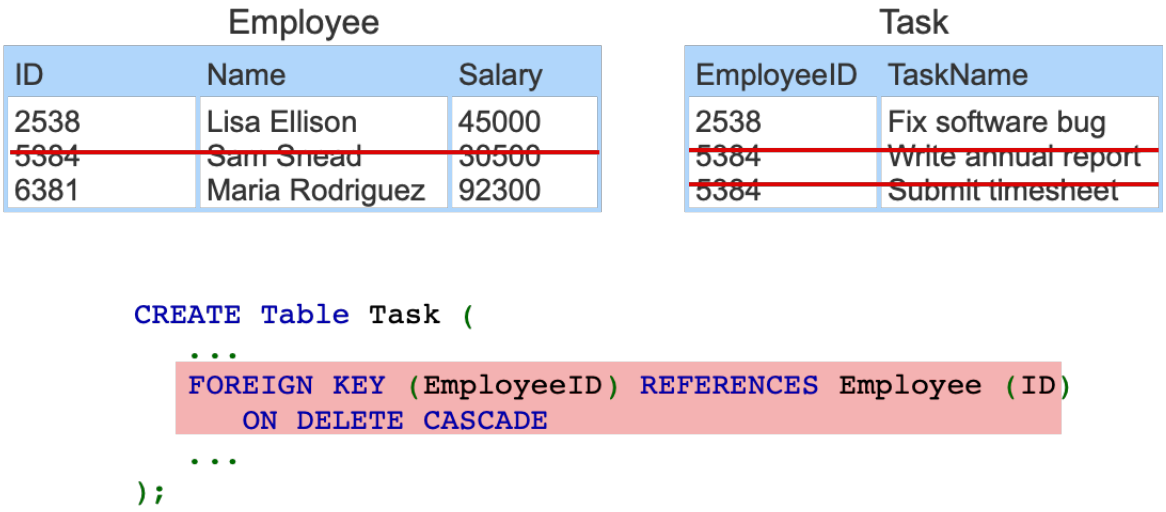
Relational rules are part of the relational model and govern data in every relational database. Ex:

- *Unique primary key*. All tables have a primary key column, or group of columns, in which values may not repeat.
- *Unique column names*. Different columns of the same table have different names.
- *No duplicate rows*. No two rows of the same table have identical values in all columns.

Business rules are based on business policy and specific to a particular database. Ex: All rows of the Employee table must have a valid entry in the DepartCode column. Ex: PassportNumber values may not repeat in different Employee rows.

Relational rules are implemented as SQL ***constraints*** and enforced by the database system. Business rules are discovered during database design and, like relational rules, often implemented as SQL constraints. However, some complex business rules must be enforced by applications running on the database.

PARTICIPATION ACTIVITY 4.1.6: Business rule example.



Animation content:

Step 1: Sam Snead has two tasks. Two tables appear named Employee and Task. Employee has three columns named ID, Name, and Salary. Task has two columns named EmployeeID and

three columns named ID, Name, and Salary. Task has two columns named EmployeeID and TaskName. Row two of Employee is highlighted with these values:
5384, Sam Snead, and 30500
Rows two and three Task are highlighted with these values:
5384, Write annual report
5384, Submit timesheet

Step 2: A business rule requires that, when an employee is deleted, the employee's tasks are also deleted. A red line strikes through the highlighted rows.

Step 3: The business rule is implemented as an SQL constraint. A CREATE TABLE statement for Task appears. Within the statement, the following clause is highlighted: FOREIGN KEY (EmployeeID) REFERENCES Employee (ID) ON DELETE CASCADE.

Animation captions:

1. Sam Snead has two tasks.
2. A business rule requires that, when an employee is deleted, the employee's tasks are also deleted.
3. The business rule is implemented as an SQL constraint.

PARTICIPATION ACTIVITY 4.1.7: Relational rules.

1) Unique primary key is an example of a relational rule.
   - True
   - False

2) Delete cascade is an example of a relational rule.
   - True
   - False

3) Data in a relational database can violate relational rules.
   - True
   - False

CHALLENGE ACTIVITY | 4.1.1: Relational model.

544874.3500394.qx3zqy7

Start

Different terms are used for similar concepts in databases, mathematics, and file systems Select the correct database term corresponding to the following terms.

Mathematics	Databases
Domain	Pick
Relation	Pick

File systems	Databases
Record	Pick
File	Pick

1 2

Check Next

Exploring further:

- Database models (Wikipedia)
- Original paper on the relational model, by E. F. Codd

4.2 Null values

NULL

NULL is a special value that represents either unknown or inapplicable data. NULL is not the same as zero for numeric data types or blanks for character data types. Ex: A zero bonus indicates an employee can, but has not, earned a bonus. A zero bonus is known and applicable, and should not be represented as NULL.

PARTICIPATION ACTIVITY | 4.2.1: NULL values in the Compensation table.

NOT NULL

Compensation

ID	Name	BirthDate	Salary	Department	Bonus
2538	Lisa Ellison	October 2, 1993	45000	Engineering	NULL
5384	Sam Snead	NULL	32000	Sales	1000
6381	Maria Rodriguez	December 21, 2001	95000	Sales	3000

Animation content:

Static figure:
The Compensation table appears, with columns ID, Name, BirthDate, Salary, Department, and Bonus. Compensation has three rows:
2538, Lisa Ellison, October 2 1993, 45000, Engineering, NULL
5384, Sam Snead, NULL, 32000, Sales, 1000
6381, Maria Rodriguez, December 21 2001, 95000, Sales, 3000
A caption NOT NULL appears above the ID column. The two NULL values in Compensation are highlighted.

Step 1: A NULL in the BirthDate column means "unknown", since all employees have a birth date. The NULL in row two of column BirthDate is highlighted.

Step 2: If Engineering employees are not paid a bonus, Lisa Ellison's NULL bonus means "inapplicable". The NULL in row one of column Bonus is highlighted.

Step 3: The ID column identifies employees and must contain valid data. The column is designated NOT NULL and cannot accept a NULL value or missing data. The caption NOT NULL appears above the ID column.

Animation captions:

1. A NULL in the BirthDate column means "unknown", since all employees have a birth date.
2. If Engineering employees are not paid a bonus, Lisa Ellison's NULL bonus means "inapplicable".
3. The ID column identifies employees and must contain valid data. The column is designated NOT NULL and cannot accept a NULL value or missing data.

PARTICIPATION ACTIVITY 4.2.2: NULL values.

Refer to the table below.

Compensation

ID	Name	Department	Salary	Bonus
2538	Lisa Ellison	Engineering	45000	0
5384	Sam Snead	Sales	30500	1000
6381	NULL	Sales	92300	3000

1) What does a NULL in the Name column represent?

- Unknown
- Inapplicable
- Either unknown or inapplicable

2) In the Bonus column of the Lisa Ellison row, what does the zero represent?

- Lisa Ellison's bonus is unknown.
- Lisa Ellison is not eligible for a bonus.
- Lisa Ellison has earned no bonus.

NOT NULL constraint

By default, columns may contain NULL values. In some cases, however, columns should never contain NULL. Ex: If a business requires that a name is specified for all employees, the Name column of an Employee table should not contain NULL.

The ***NOT NULL*** constraint prevents a column from having a NULL value. Statements that insert NULL, or update a value to NULL, are automatically rejected. NOT NULL follows the column name and data type in a CREATE TABLE statement.

PARTICIPATION ACTIVITY 4.2.3: NOT NULL constraint.

```
CREATE TABLE Employee (
   ID          SMALLINT UNSIGNED,
   Name        VARCHAR(60) NOT NULL,
   BirthDate   DATE,
   Salary      DECIMAL(7,2)
);
```

Employee

ID	Name	BirthDate	Salary
6381	Maria Rodriguez	NULL	92300
~~2538~~	~~NULL~~	~~1990-12-03~~	~~423.0~~

Animation content:

Static figure:
This query appears:
Begin SQL code:
CREATE TABLE Employee (
ID SMALLINT UNSIGNED,
Name VARCHAR(60) NOT NULL,
BirthDate DATE,
Salary DECIMAL(7,2)
);
End SQL code.

The Employee table appears, with columns ID, Name, BirthDate, and Salary. Employee has two rows:
6381, Maria Rodriguez, NULL, 92300
2538, NULL, 1990-03, 423.00
The second row is struck through with a red line.

Step 1: The BirthDate column allows NULL values by default. The query clause BirthDate DATE is highlighted. The first row of Employee appears.

Step 2: The NOT NULL constraint prevents Name from being NULL when inserting a new row into Employee. The query keywords NOT NULL are highlighted. The second row of Employee appears. NULL is highlighted in this row. The row is struck through with a red line.

Animation captions:

1. The BirthDate column allows NULL values by default.
2. The NOT NULL constraint prevents Name from being NULL when inserting a new row into

Employee.

PARTICIPATION ACTIVITY 4.2.4: NOT NULL constraint.

Refer to the statement below.

```
CREATE TABLE Department (
   Code      TINYINT UNSIGNED NOT NULL,
   Name      VARCHAR(20),
   ManagerID SMALLINT
);
```

1) Which columns may contain NULL values?

- Code
- Code and Name
- Name and ManagerID

2) Which alteration to the CREATE TABLE statement prevents ManagerID from being NULL?

- `ManagerID NOT NULL SMALLINT`
- `ManagerID NOT NULL`
- `ManagerID SMALLINT NOT NULL`

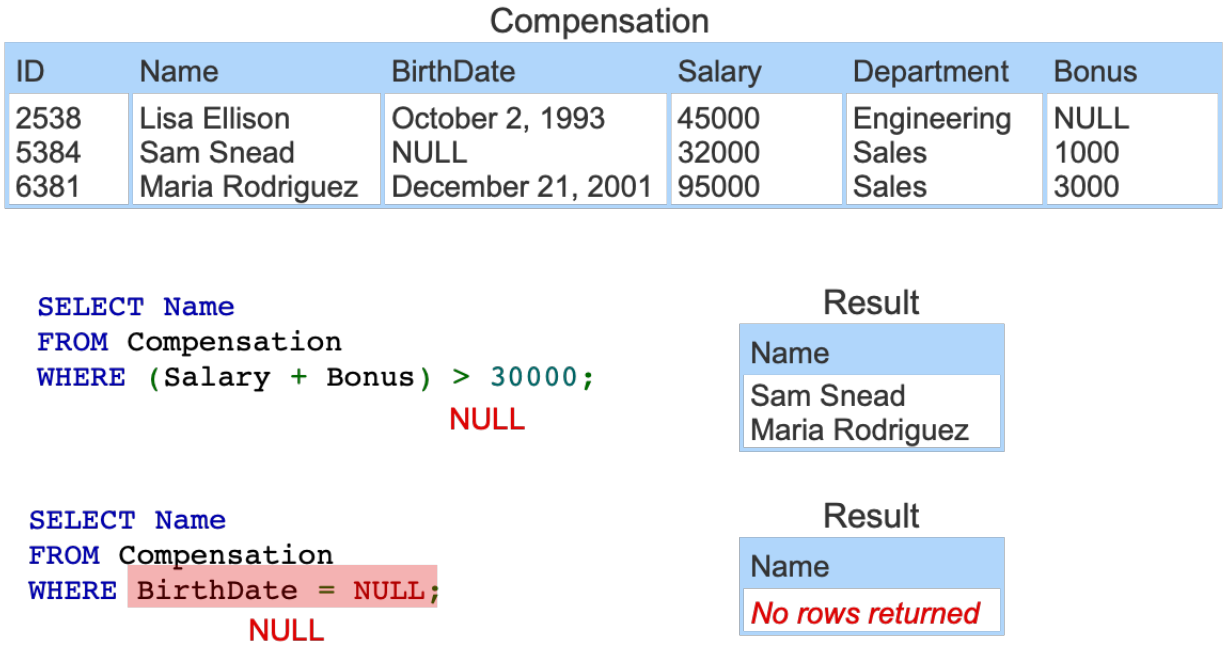
3) What happens when a user attempts to insert a new department without a Code value?

- The database accepts the insert and assigns Code with zero.
- The database accepts the insert and assigns Code with NULL.
- The database rejects the insert.

NULL arithmetic and comparisons

When arithmetic or comparison operators have one or more NULL operands, the result is NULL. When a WHERE clause evaluates to NULL for values in a row, the row is not selected.

PARTICIPATION ACTIVITY 4.2.5: NULL arithmetic and comparisons.



Animation content:

Static figure:
The Compensation table appears, with columns ID, Name, BirthDate, Salary, Department, and Bonus. Compensation has three rows:
2538, Lisa Ellison, October 2 1993, 45000, Engineering, NULL
5384, Sam Snead, NULL, 32000, Sales, 1000
6381, Maria Rodriguez, December 21 2001, 95000, Sales, 3000

A query appears:
Begin SQL code:
SELECT Name

SELECT Name
FROM Compensation
WHERE (Salary + Bonus) > 3000;
End SQL code.
NULL appears under the expression in the WHERE clause. The result appears next to this query, with column Name and two rows:
Sam Snead
Maria Rodriguez

A second query appears.
Begin SQL code:
SELECT Name
FROM Compensation
WHERE BirthDate = NULL;
End SQL code.
NULL appears under the expression in the WHERE clause. The result appears next to this query, with column Name and no rows.

Step 1: Lisa Ellison's Bonus is NULL. As a result, Salary + Bonus is NULL and (Salary + Bonus) > 30000 is NULL. The first query appears. The WHERE clause evaluates to NULL for the first row of Compensation.

Step 2: The SELECT statement does not select a name from a row when the WHERE clause is NULL, so Lisa Ellison is not selected. The first result table appears without Lisa Ellison.

Step 3: The = comparison operator returns NULL when either or both operands are NULL, so the WHERE clause evaluates to NULL for Sam Snead. The second query appears. NULL appears under the WHERE clause. The second result table appears with no rows.

Animation captions:

1. Lisa Ellison's Bonus is NULL. As a result, Salary + Bonus is NULL and (Salary + Bonus) > 30000 is NULL.
2. The SELECT statement does not select a name from a row when the WHERE clause is NULL, so Lisa Ellison is not selected.
3. The = comparison operator returns NULL when either or both operands are NULL, so the WHERE clause evaluates to NULL for Sam Snead.

PARTICIPATION ACTIVITY 4.2.6: NULL arithmetic and comparisons.

Refer to the table below.

Compensation

ID	Name	Salary	Bonus
2538	Lisa Ellison	45000	NULL
5348	Sam Snead	32000	32000
6381	Maria Rodriguez	95000	98000
8820	Jiho Chen	NULL	NULL

What name is selected by each statement?

1)
```
SELECT Name
FROM Compensation
WHERE Salary = Bonus;
```

Check Show answer

2)
```
SELECT Name
FROM Compensation
WHERE (Salary / Bonus) <
1.0;
```

Check Show answer

IS NULL operator

Since comparison operators return NULL when either operand is NULL, comparison operators cannot be used to select NULL values. Ex:
`SELECT * FROM Employee WHERE Salary = NULL;` never returns any rows, because the WHERE clause is always NULL.

Instead, the ***IS NULL*** and ***IS NOT NULL*** operators must be used to select NULL values.
`Value IS NULL` returns TRUE when the value is NULL. `Value IS NOT NULL` returns TRUE when the value is not NULL.

PARTICIPATION ACTIVITY 4.2.7: Selecting NULL values.

Country

Code	Name	HeadOfState	IndepYear	Population
ABW	Aruba	Beatrix	NULL	103000
AIA	Anguilla	Charles III	1920	NULL
AFG	Afghanistan	Mohammad Omar	1919	22720000
AGO	Angola	Jose dos Santos	1975	12878000

```
SELECT *
FROM Country
WHERE IndepYear IS NULL;
```

Code	Name	HeadOfState	IndepYear	Population
ABW	Aruba	Beatrix	NULL	103000

```
SELECT *
FROM Country
WHERE Population IS NOT NULL;
```

Code	Name	HeadOfState	IndepYear	Population
ABW	Aruba	Beatrix	NULL	103000
AFG	Afghanistan	Mohammad Omar	1919	22720000
AGO	Angola	Jose dos Santos	1975	12878000

Animation content:

Static figure:
The Country table appears with columns Code, Name, HeadOfState, IndepYear, and Population.
Country has four rows:
ABW, Aruba, Beatrix, NULL, 103000
AIA, Anguilla, Charles III, 1920, NULL
AFG, Afghanistan, Mohammad Omar, 1919, 22720000
AGO, Angola, Jose dos Santos, 1975, 12878000

A query appears:
Begin SQL code:
SELECT *
FROM Country
WHERE IndepYear IS NULL;
End SQL code.
The result table appears next to the query, with the same columns as Country. The result has one row:
ABW, Aruba, Beatrix, NULL, 103000

A second query appears:

A second query appears.
Begin SQL code:
SELECT *
FROM Country
WHERE Population IS NOT NULL;
End SQL code.
A second result table appears next to the second query, with the same columns as Country. The result has three rows:
ABW, Aruba, Beatrix, NULL, 103000
AFG, Afghanistan, Mohammad Omar, 1919, 22720000
AGO, Angola, Jose dos Santos, 1975, 12878000

Step 1: The NULL in column IndepYear represents inapplicable data - a country has not achieved independence. The NULL in column Population represents unknown data. The Country table appears.

Step 2: Selecting rows where IndepYear IS NULL returns in one row. The first query appears. The first result table appears.

Step 3: Selecting rows where Population IS NOT NULL returns three rows. The second query appears. The second result table appears.

Animation captions:

1. The NULL in column IndepYear represents inapplicable data - a country has not achieved independence. The NULL in column Population represents unknown data.
2. Selecting rows where IndepYear IS NULL returns in one row.
3. Selecting rows where Population IS NOT NULL returns three rows.

PARTICIPATION ACTIVITY 4.2.8: Selecting NULL values.

Refer to the table below.

Country

Code	Name	HeadOfState	IndepYear	Population
ABW	Aruba	Beatrix	NULL	103000
AIA	Anguilla	Charles III	1920	NULL
AFG	Afghanistan	Mohammad Omar	1919	22720000
AGO	Angola	Jose dos Santos	1975	12878000

1) How many rows are in the result of the following query?

SELECT *
FROM Country
WHERE IndepYear IS NULL;

1) How many rows are returned?

```
SELECT *
FROM Country
WHERE Population = NULL;
```

- 0
- 1
- 3

2) How many rows are returned?

```
SELECT *
FROM Country
WHERE Population IS NULL;
```

- 0
- 1
- 3

3) What is missing to select all rows except Aruba?

```
SELECT *
FROM Country
WHERE IndepYear _____;
```

- `!= NULL`
- `IS NULL`
- `IS NOT NULL`

PARTICIPATION ACTIVITY 4.2.9: Select songs with NULL values.

This activity failed to load. Please try refreshing the page. If that fails, you might also try clearing your browser's cache.

If an issue persists,

send feedback to zyBooks support

NULL logic

NULL logic

In traditional mathematical logic, expressions are always TRUE or FALSE. When NULL is present, however, a logical expression may be either TRUE, FALSE, or NULL. NULL indicates the value of a logical expression is uncertain. Ex:

- TRUE AND TRUE is TRUE.
- TRUE AND FALSE is FALSE.
- TRUE AND NULL is NULL.

The value of logical expressions containing NULL operands is defined in ***truth tables***.

Figure 4.2.1: MySQL truth tables.

x	y	x AND y	x OR y
TRUE	NULL	NULL	TRUE
NULL	TRUE		
FALSE	NULL	FALSE	NULL
NULL	FALSE		
NULL	NULL	NULL	NULL

x	NOT x
NULL	NULL

MySQL does not have a special data type for logical values. Internally, MySQL represents FALSE as 0 and TRUE as 1. In query results, FALSE and TRUE are also displayed as 0 and 1.

Since null logic is not standardized in mathematics, implementation details vary. Ex: Oracle Database displays a NULL logical expression as UNKNOWN.

PARTICIPATION ACTIVITY 4.2.10: NULL logic.

Refer to the Compensation table below.

Compensation

ID	Name	Salary	Bonus

ID	Name	Salary	Bonus
2538	Lisa Ellison	115000	NULL
5348	Sam Snead	35000	55000
6381	Maria Rodriguez	95000	3000

In MySQL, what names are selected by the following queries?

1)
```
SELECT Name
FROM Compensation
WHERE Salary > 30000 OR Bonus
> 1000;
```

- Lisa Ellison
- Sam Snead and Maria Rodriguez
- Lisa Ellison, Sam Snead, and Maria Rodriguez

2)
```
SELECT Name
FROM Compensation
WHERE Salary > 30000 AND BONUS
> 1000;
```

- Lisa Ellison
- Sam Snead and Maria Rodriguez
- Lisa Ellison, Sam Snead, and Maria Rodriguez

3)
```
SELECT Name
FROM Compensation
WHERE NOT(Salary > 30000 AND
BONUS > 1000);
```

- No names are selected
- Lisa Ellison
- Lisa Ellison, Sam Snead, and Maria Rodriguez

PARTICIPATION ACTIVITY 4.2.11: Select songs with NULL title or artist.

This activity failed to load. Please try refreshing the

page. If that fails, you might also try clearing your browser's cache.

If an issue persists,

send feedback to zyBooks support

CHALLENGE ACTIVITY | 4.2.1: Null values.

544874.3500394.qx3zqy7

Start

```
CREATE TABLE Country (
   ISOCode3 CHAR(3) NOT NULL,
   Population INTEGER,
   Area DECIMAL(9, 2),
   Over65PopPct FLOAT NOT NULL,
   ISOCode2 CHAR(2),
   Name VARCHAR(15)
);
```

Which columns can contain NULL values?

- [ ] ISOCode3
- [ ] Population
- [ ] Area
- [ ] Over65PopPct
- [ ] ISOCode2
- [ ] Name

1 2 3 4

Check Next

Exploring further:

- MySQL null values

4.3 Primary keys

Primary keys

A ***primary key*** is a column, or group of columns, used to identify a row. To ensure that each value identifies exactly one row, a primary key must be unique and not NULL.

In table diagrams, a bullet (●) precedes the primary key. The primary key is usually the left-most column of a table, but the position is not significant to the database. Ex: ID is the primary key of the Employee table below.

Table 4.3.1: Employee table with primary key.

Employee

● ID	Name	Salary
2538	Lisa Ellison	45000
5384	Sam Snead	30500
6381	Maria Rodriguez	92300

Often, primary key values are used in the WHERE clause to select a specific row.

Figure 4.3.1: Primary key used to select a specific row.

```
SELECT Name
FROM Employee
WHERE ID = 5384;
```

```
Sam Snead
```

PARTICIPATION ACTIVITY 4.3.1: Primary keys.

Refer to the Employee table:

● ID	Name	Salary
2538	Lisa Ellison	45000

5384	Sam Snead	30500
6381	Maria Rodriguez	92300

1) Name is a good primary key.

- True
- False

2) A new employee can be added without an ID value.

- True
- False

3) A new employee can be added with ID 5384.

- True
- False

Composite primary keys

Sometimes multiple columns are necessary to identify a row. A ***simple primary key*** consists of a single column. A ***composite primary key*** consists of multiple columns. A composite primary key must be:

- *Unique*. Values of primary key columns, when grouped together, must be unique. No group of values may repeat in multiple rows.
- *Not NULL*. No column of a composite primary key may contain a NULL value.
- ***Minimal***. All primary key columns are necessary for uniqueness. When any column is removed, the resulting simple or composite column is no longer unique.

Simple primary keys are necessarily minimal, since no column can be removed from a simple key.

In text, composite primary keys are enclosed in parentheses. Ex: (ColumnA, ColumnB). In table diagrams, a bullet (•) precedes every column of a composite primary key.

PARTICIPATION ACTIVITY 4.3.2: Composite primary keys.

Composite

primary key

Family

• ID	• Number	Relationship	Name
2538	1	Spouse	Henry Ellison
2538	2	Son	Edward Ellison
6381	1	Spouse	Jose Rodriguez
6381	2	Daughter	Gina Rodriguez
6381	3	Daughter	Clara Rodriguez

Animation content:

Step 1: ID is not unique in the Family table, since one employee may have several family members. A table Family appears with columns ID, Number, Relationship, and Name. Rows one and two of column ID are highlighted and both contain the value 2538. Rows three, four, and five of column ID are highlighted and all contain the value 6381.

Step 2: ID and Number together is unique, so (ID, Number) is a composite primary key. Row one of columns ID and Number are highlighted and contain values 2583 and 1 respectively. Row two of columns ID and Number is highlighted and contains the values 2583 and 2 respectively. Row three of columns ID and Number is highlighted and contains the values of 6381 and 1 respectively. Row four of columns ID and Number is highlighted and contains the values 6381 and 2 respectively. Row five of columns ID and Number is highlighted contains the values 6381 and 3 respectively.

Step 3: (ID, Number, Relationship) is unique. However, the Relationship column is unnecessary, so (ID, Number, Relationship) is not minimal. Row one of columns ID Number and Relationship is highlighted and contains the values 2583 1 and Spouse respectively. Row two of columns ID Number and Relationship is highlighted and contains the values 2583 2 and Son respectively. Row three of columns ID Number and Relationship is highlighted contains the values 6381 1 and Spouse respectively. Row four of columns ID Number and Relationship is highlighted and contains the values 6381 2 and Daughter respectively. Row five of columns ID Number and Relationship is highlighted and contains the values 6381 3 and Daughter respectively.

Animation captions:

1. ID is not unique in the Family table, since one family may have several family members.
2. ID and Number together is unique, so (ID, Number) is a composite primary key.
3. (ID, Number, Relationship) is unique. However, the Relationship column is unnecessary, so (ID, Number, Relationship) is not minimal.

PARTICIPATION ACTIVITY 4.3.3: Composite primary keys.

Refer to the tables below.

Family

• ID	• Number	Relationship	Name
2538	1	Spouse	Henry Ellison
2538	2	Son	Edward Ellison
6381	1	Spouse	Jose Rodriguez
6381	2	Daughter	Gina Rodriguez
6381	3	Daughter	Clara Rodriguez

PhoneNumber

AreaCode	Exchange	Number
510	899	1111
212	899	1111
510	899	1234
212	611	1111

1) Can (ID, Relationship) be the primary key of Family?

- Yes
- No
- Cannot determine answer from data in the table.

2) The primary key of the PhoneNumber table is not indicated with a bullet. What is the primary key of PhoneNumber?

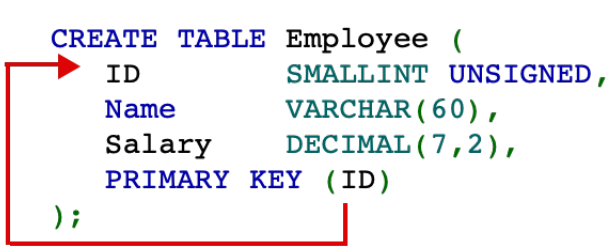
- Table has no primary key
- (AreaCode, Number)
- (AreaCode, Exchange, Number)

PRIMARY KEY constraint

The ***PRIMARY KEY*** constraint in a CREATE TABLE statement names the table's primary key. This constraint ensures that a column or group of columns is always unique and non-null.

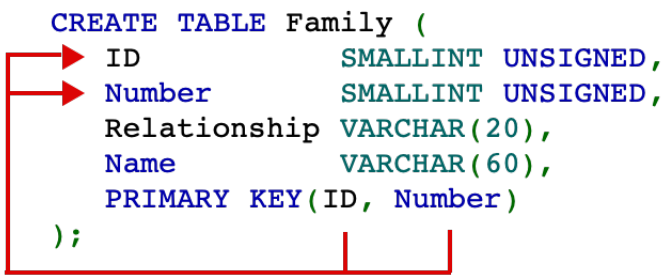
In a CREATE TABLE statement, the primary key column definition usually appears first, followed by other column definitions and the primary key constraint. However, the order of CREATE TABLE clauses is not significant.

PARTICIPATION ACTIVITY 4.3.4: Adding primary key constraints to tables.



Employee

ID	Name	Salary
2538	Lisa Ellison	45000
5384	Sam Snead	30400
6381	Maria Rodriguez	92300



Family

ID	Number	Relationship	Name
2538	1	Spouse	Henry Ellison
2538	2	Son	Edward Ellison
6381	1	Spouse	Jose Rodriguez
6381	2	Daughter	Gina Rodriguez
6381	3	Daughter	Clara Rodriguez

Animation content:

Static figure:
An SQL statement appears.
Begin SQL code:
CREATE TABLE Employee (
  ID SMALLINT UNSIGNED,
  Name VARCHAR(60),
  Salary DECIMAL(7,2),
  PRIMARY KEY (ID)
);
End SQL code.

An Employee table appears to the right of this statement, with columns ID, Name, and Salary. The table has three rows.

A second SQL statement appears.
Begin SQL code:
CREATE TABLE Family (
  ID SMALLINT UNSIGNED,
  Number SMALLINT UNSIGNED,
  Relationship VARCHAR(20),
  Name VARCHAR(60),
  PRIMARY KEY(ID, Number)

);
End SQL code.

A Family table appears to the right of this statement, with columns ID, Number, Relationship, and Name. The table has five rows.

Step 1: The CREATE TABLE statement uses the keywords PRIMARY KEY to indicate the ID column is the table's primary key. The first SQL statement appears. The clause PRIMARY KEY (ID) is highlighted. The Employee table appears with a bullet next to the ID column. Employee has no rows.

Step 2: All rows added to the Employee table must have a unique ID. Three rows are added to Employee. All rows have different values in the ID column.

Step 3: The PRIMARY KEY constraint identifies the ID and Number columns as the Family table's composite primary key. The second SQL statement appears. The clause PRIMARY KEY (ID, Number) is highlighted. The Family table appears with bullets before columns ID and Number. Family has no rows.

Step 4: All rows added to the Family table must have a unique combination of ID and Number. Five rows are added to Family. All rows have different values in the composite column (ID, Number):
2538, 1
2538, 2
6381, 1
6381, 2
6381, 3

Animation captions:

1. The CREATE TABLE statement uses the keywords PRIMARY KEY to indicate the ID column is the table's primary key.
2. All rows added to the Employee table must have a unique ID.
3. The PRIMARY KEY constraint identifies the ID and Number columns as the Family table's composite primary key.
4. All rows added to the Family table must have a unique combination of ID and Number.

PARTICIPATION ACTIVITY | 4.3.5: PRIMARY KEY constraints.

Refer to the animation above.

1) Lisa, Sam, and Maria must have unique IDs and names.

- True
- False

2) The PRIMARY KEY constraint may include multiple columns.

- True
- False

3) Assuming the Family table has the five rows shown above, a new row with values (2538, 2, 'Daughter', 'Ella Ellison') may be added to the Family table.

- True
- False

Auto-increment columns

An ***auto-increment column*** is a numeric column that is assigned an automatically incrementing value when a new row is inserted. The ***AUTO_INCREMENT*** keyword defines an auto-increment column. AUTO_INCREMENT follows the column's data type in a CREATE TABLE statement.

Integer primary keys are commonly implemented as auto-increment columns. In MySQL, AUTO_INCREMENT may be applied only to primary key columns.

Figure 4.3.2: Auto-increment primary key example.

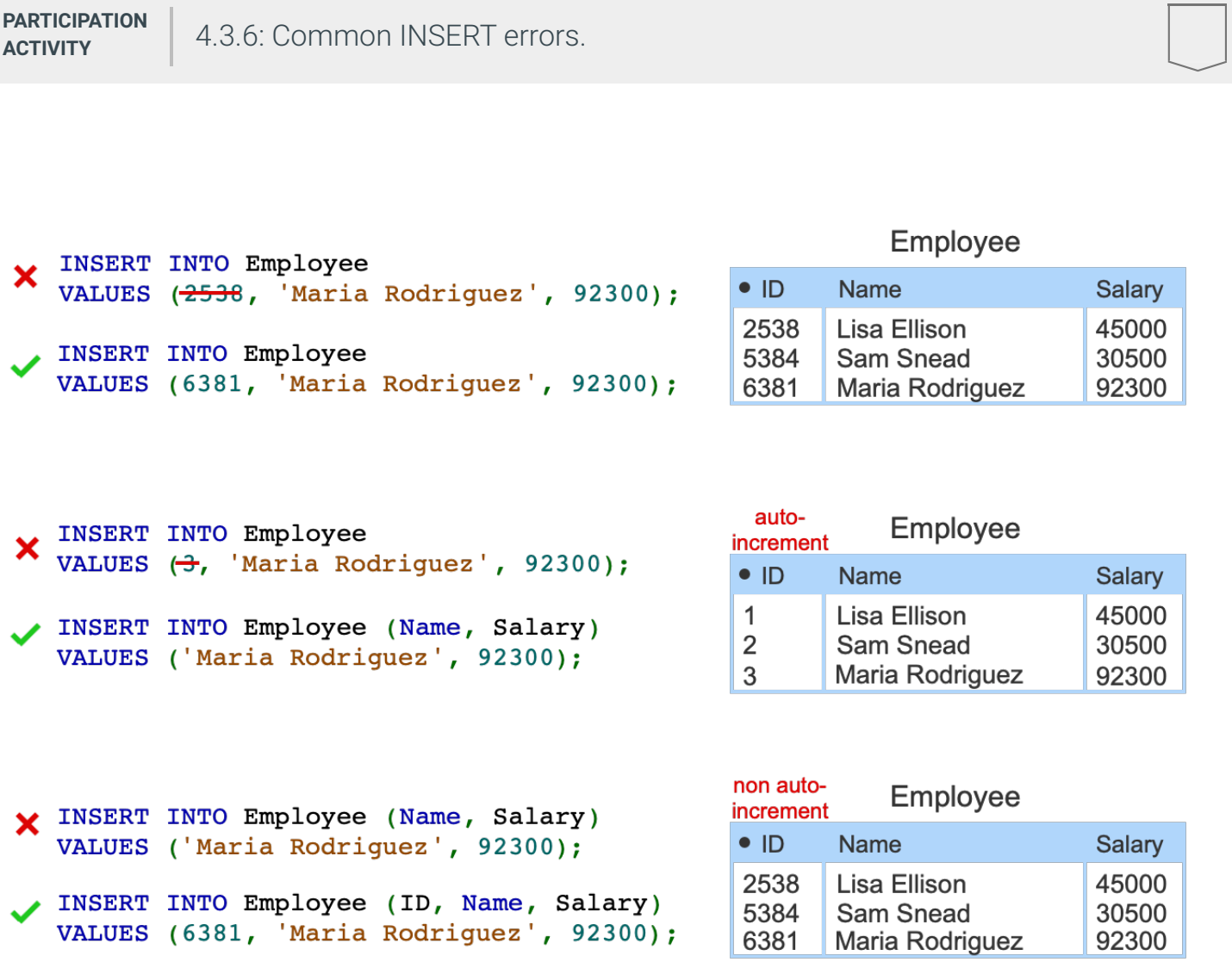
```
CREATE TABLE Employee (
   ID        SMALLINT UNSIGNED
AUTO_INCREMENT,
   Name      VARCHAR(60),
   BirthDate DATE,
   Salary    DECIMAL(7,2),
   PRIMARY KEY (ID)
);
```

Database users occasionally make the following errors when inserting primary keys:

- Inserting values for auto-increment primary keys.
- Omitting values for primary keys that are not auto-increment columns.

MySQL allows insertion of a specific value to an auto-increment column. However, overriding auto-increment for a primary key is usually a mistake.

PARTICIPATION ACTIVITY 4.3.6: Common INSERT errors.



Animation content:

Step 1: The INSERT statement uses an ID that already exists in Employee. Duplicate primary key values cannot be added, so a unique ID must be chosen. A table Employee appears, with columns ID, Name, and Salary. ID is a primary key. There are two lines of code. The first line of code states INSERT INTO Employee. Line two of code states VALUES left parenthesis 2538 comma apostrophe Maria Rodriguez apostrophe comma 92300 right parenthesis semicolon. 2538 in the second line of code is boxed and value 2538 in column ID of table Employee is

2538 in the second line of code is boxed and value 2538 in column ID of table Employee is highlighted. 2538 in the second line of code is crossed out and a red X is put next to these two lines of code. Two new lines of code appear. The first line of code states left parenthesis 6381 comma apostrophe Maria Rodriguez apostrophe comma 92300 right parenthesis semicolon. 6381 in the second line of code is boxed and values 6381 Maria Rodriguex and 92300 are added as a new row to columns ID Name and Salary respectively. A green checkmark is placed next to the code.

Step 2: If ID is an auto-increment column, the ID should not be listed in the INSERT statement. The database assigns the ID automatically. A second table Employee appears, with columns ID, Name, and Salary. ID is a primary key and is also labeled auto-increment. Two new lines of code appear. The first line one code states INSERT INTO Employee. The second line of code states VALUES left parenthesis 3 comma apostrophe Maria Rodriguex apostrophe comma 92300 right parenthesis semicolon. The 3 in the second line of code is boxed and then crossed out. A big red X appears next to these two lines of code. Two new lines of code appear. The first line of code states INSERT INTO Employee left parenthesis Name comma Salary right parenthesis. The second line of code states VALUES left parenthesis apostrophe Maria Rodriguex apostrophe comma 92300 right parenthesis semicolon. The values Maria Rodriguez and 92300 are added into a new row in columns Name and Salary of table Employee respectively. 3 is filled into the same row in column ID and is highlighted. A green check mark appears next to the two lines of code.

Step 3: If ID is not an auto-increment column, then an ID value must be specified. A third Employee table appears, with columns ID, Name, and Salary. ID is a primary key and is also labeled non auto-increment, and Salary is labeled NOT NULL. Two new lines of code appear. The first line one code states INSERT INTO Employee left parenthesis Name comma Salary right parenthesis. The second line of code states VALUES left parenthesis apostrophe Maria Rodriguez apostrophe comma 92300 right parenthesis semicolon. A big red X appears next to these two lines of code. Two new lines of code appear. The first line of code states INSERT INTO Employee left parenthesis ID comma Name comma Salary right parenthesis. The second line of code states VALUES left parenthesis 6381 comma apostrophe Maria Rodriguez apostrophe comma 92300 right parenthesis semicolon. The values 6381 Maria Rodriguez and 92300 are added into a new row in columns ID Name and Salary of table Employee respectively. A green check mark appears next to the two lines of code.

Animation captions:

1. The INSERT statement uses an ID that already exists in Employee. Duplicate primary key values cannot be added, so a unique ID must be chosen.
2. If ID is an auto-increment column, the ID should not be listed in the INSERT statement. The database assigns the ID automatically.
3. If ID is not an auto-increment column, then an ID value must be specified.

PARTICIPATION ACTIVITY 4.3.7: Insert rows into Movie table.

This activity failed to load. Please try refreshing the page. If that fails, you might also try clearing your browser's cache.

If an issue persists,

send feedback to zyBooks support

PARTICIPATION ACTIVITY 4.3.8: Common INSERT errors.

Refer to the table definition below.

```
CREATE TABLE Department (
  Code TINYINT UNSIGNED AUTO_INCREMENT,
  Name VARCHAR(20) NOT NULL,
  ManagerID SMALLINT UNSIGNED,
  PRIMARY KEY (Code)
);
```

1) Which statement correctly inserts Engineering?

- ```
  INSERT INTO Department
  (Code, Name, ManagerID)
  VALUES (44,
  'Engineering', 2538);
  ```
- ```
  INSERT INTO Department
  VALUES ('Engineering',
  2538);
  ```
- ```
  INSERT INTO Department
  (Name, ManagerID)
  VALUES ('Engineering',
  2538);
  ```

2) Which statement correctly inserts an unnamed department with no manager?

```
INSERT INTO Department
```
```

- INSERT INTO Department
  (Name, ManagerID)
  VALUES ('', NULL);

- INSERT INTO Department
  VALUES (NULL, '', NULL);

- INSERT INTO Department
  (Name, ManagerID)
  VALUES ('');

CHALLENGE ACTIVITY | 4.3.1: Primary keys.

544874.3500394.qx3zqy7

Start

Country

ISOCode2	CountryName	Capital	ContinentCode
HU	Hungary	Budapest	EU
ID	Indonesia	Jakarta	AS
GN	Guinea	Conakry	AF
US	United States	Washington	NA
TV	Tuvalu	Funafuti	OC

```
SELECT CountryName
FROM Country
WHERE ISOCode2 = 'HU';
```

What is returned?

- [ ] Hungary
- [ ] Indonesia
- [ ] Guinea
- [ ] United States
- [ ] Tuvalu

1 2 3 4 5

Check Next

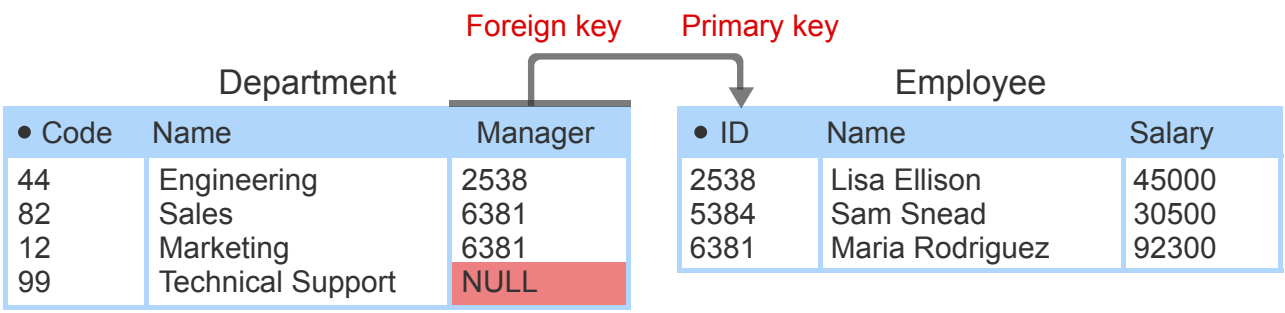
4.4 Foreign keys

Foreign keys

A ***foreign key*** is a column, or group of columns, that refer to a primary key. The data types of the foreign and primary keys must be the same, but the names may be different. Unlike primary keys, foreign key values may be NULL and are not necessarily unique. However, a foreign key value that is not NULL must match some value of the referenced primary key.

In table diagrams, an arrow indicates a foreign key. The arrow starts at the foreign key and points to the table containing the referenced primary key.

PARTICIPATION ACTIVITY 4.4.1: The foreign key Manager refers to the primary key ID.



Animation content:

Step 1: The Department table's column is a foreign key that refers to the primary key ID in the Employee table. The Department table is on the left and the Employee table is on the right. Department has columns Code, Name, and Manager. Code is preceded by a solid circle. Department has four rows with values (44, Engineering, 2538), (82, Sales, 6381), (12, Marketing, 6381), and (99, Technical Support, NULL). Employee has columns ID, Name, and Salary. ID is preceded by a solid circle. Employee has three rows, with values (2538, Lisa Ellison, 45000), (5384, Sam Snead, 30500), and (6381, Maria Rodriguez, 92300). Manager is labeled Foreign key, ID is labeled Primary key, and an arrow points from Manager to ID.

Step 2: Lisa Ellison manages the engineering department. The value 2538 in row one of Department Manager, and in row one of Employee ID, is highlighted.

Step 3: Maria Rodriguez manages the sales and marketing departments. The value 6381 in rows

two and three of Department Manager, and row three of Employee ID, is highlighted.

Step 4: The Technical Support department has no manager. The value NULL in row four Department Manager is highlighted.

Animation captions:

1. The Department table's Manager column is a foreign key that refers to the primary key ID in the Employee table.
2. Lisa Ellison manages the engineering department.
3. Maria Rodriguez manages the sales and marketing departments.
4. The Technical Support department has no manager.

PARTICIPATION ACTIVITY 4.4.2: Foreign keys.

Refer to the tables above.

1) The data type of Manager and ID must be the same.
   - True
   - False

2) NULL in the Manager column refers to an Employee row with a NULL ID.
   - True
   - False

3) Sam Snead does not manage a department.
   - True
   - False

4) Replacing NULL in the Manager column with 5384 assigns Sam Snead as the manager of the Technical Support department.
   - True
   - False

5) The NULL in the Manager column may be replaced with 9876.

- True
- False

6) Values in a foreign key must be unique.

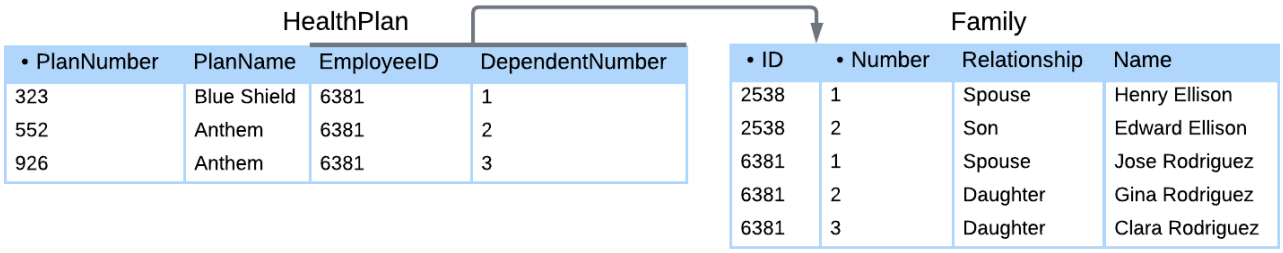
- True
- False

Composite foreign keys

A foreign key that refers to a composite primary key must also be composite. All columns of a composite foreign key value must either be NULL or match some primary key value. In table diagrams, the tail of the arrow extends across all columns of a composite foreign key.

In the figure below, the composite foreign key (EmployeeID, DependentNumber) of HealthPlan refers to the composite primary key ((ID, Number) of Family.

Figure 4.4.1: Composite foreign keys.



• PlanNumber	PlanName	EmployeeID	DependentNumber
323	Blue Shield	6381	1
552	Anthem	6381	2
926	Anthem	6381	3

• ID	• Number	Relationship	Name
2538	1	Spouse	Henry Ellison
2538	2	Son	Edward Ellison
6381	1	Spouse	Jose Rodriguez
6381	2	Daughter	Gina Rodriguez
6381	3	Daughter	Clara Rodriguez

PARTICIPATION ACTIVITY 4.4.3: Composite foreign keys.

Refer to the tables above.

1) Which family member has the Blue Shield health plan?

- Harry Ellison
- Jose Rodriguez
- Clara Rodriguez

- Clara Rodriguez

2) Can the HealthPlan table contain the value (2538, NULL) in (EmployeeID, DependentNumber)?

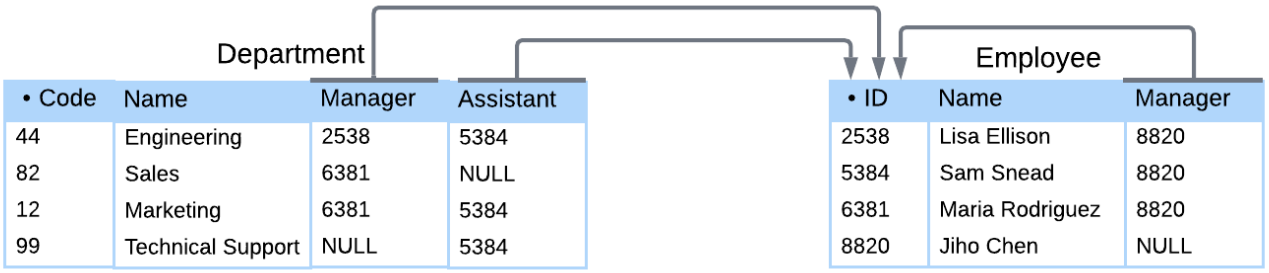
- Yes, with the Family data shown above
- Yes, if Family had additional rows and primary key values
- No

Special cases

Multiple foreign keys may refer to the same primary key. A foreign key may refer to the primary key of the same table.

In the figure below, the Manager and Assistant foreign keys of Department both refer to ID, the primary key of Employee. The Manager foreign key of Employee also refers to the primary key of Employee.

Figure 4.4.2: Special cases.



PARTICIPATION ACTIVITY 4.4.4: Special cases.

Refer to the tables above.

1) Which department has the same manager and assistant?

- Engineering
- Sales
- No department has the same

- No department has the same manager and assistant.

2) Who is Lisa Ellison's manager?

- Sam Snead
- Maria Rodriguez
- Jiho Chen

Foreign key constraint

A foreign key constraint is created with a foreign key clause in the CREATE TABLE statement. The clause consists of the ***FOREIGN KEY*** keyword followed by the foreign key column, and the ***REFERENCES*** keyword followed by the referenced table and primary key column. The clause may appear anywhere in the CREATE TABLE statement, but usually follows all column definitions.

When a foreign key constraint is specified, the database rejects insert, update, and delete statements that violate referential integrity.

Figure 4.4.3: FOREIGN KEY syntax.

```
CREATE TABLE TableName (
  . . .
  FOREIGN KEY (ColumnName) REFERENCES TableName
(ColumnName),
  . . .
);
```

PARTICIPATION ACTIVITY 4.4.5: Foreign key constraint.

Department

• Code	Name	ManagerID
44	Engineering	2538
82	Sales	6381
12	Marketing	~~9999~~ 6381
99	Technical Support	7343

Employee

• ID	Name	BirthDate	Salary
2538	Lisa Ellison	1993-10-02	45000
5384	Sam Snead	1995-03-15	30500
6381	Maria Rodriguez	2001-12-21	92300
7343	Gary Smith	1984-09-22	85000

```
CREATE TABLE Department (
   Code       TINYINT UNSIGNED,
   Name       VARCHAR(20),
```

```
    ManagerID SMALLINT UNSIGNED,
    PRIMARY KEY (Code),
    FOREIGN KEY (ManagerID) REFERENCES Employee(ID)
);
```

Animation content:

Static figure:
Two tables appear. The Department table has columns Code, Name, and ManagerID. Code has a bullet. Department has four rows. The Employee table has columns ID, Name, BirthDate, and Salary. ID has a bullet. Employee has four rows. An arrow points from the ManagerID column to the ID column.

An SQL statement appears.
Begin SQL code:
CREATE TABLE Department (
  Code TINYINT UNSIGNED,
  Name VARCHAR(20),
  ManagerID SMALLINT UNSIGNED,
  PRIMARY KEY (Code),
  FOREIGN KEY (ManagerID) REFERENCES Employee(ID)
);
End SQL code.

Step 1: The Employee table has primary key ID. The Employee table appears.

Step 2: The Department table is created with a FOREIGN KEY constraint that REFERENCES the Employee ID column. The FOREIGN KEY clause of the SQL statement is highlighted.The Department table appears with no rows. The arrow from ManagerID to ID appears.

Step 3: When rows are added to Department, the ManagerID value must exist in the ID column. 9999 does not appear in ID and is rejected. Four rows are added to the Department table:
Row one of column ManagerID has value 2538 and matches the value 2538 in the column ID of table Employee.
Row two of column ManagerID has value 6381 and matches the value 6381 in row three of column ID of table Employee.
Row three of column ManagerID has value 9999 and does not match any value in column ID of table Employee. Value 9999 is crossed out and changed to 6381, which matches the value in row three of column ID of table Employee.
Row four of column ManagerID has value 7343 and matches the value 7343 of column ID of

table Employee.

Animation captions:

1. The Employee table has primary key ID.
2. The Department table is created with a FOREIGN KEY constraint that REFERENCES the Employee ID column.
3. When rows are added to Department, the ManagerID value must exist in the ID column. 9999 does not appear in ID and is rejected.

PARTICIPATION ACTIVITY 4.4.6: Add primary and foreign key constraints.

This activity failed to load. Please try refreshing the page. If that fails, you might also try clearing your browser's cache.

If an issue persists,

send feedback to zyBooks support

PARTICIPATION ACTIVITY 4.4.7: Foreign key constraint.

1) In a CREATE TABLE statement, the FOREIGN KEY constraint must follow all column declarations.
   - True
   - False

2) In a FOREIGN KEY constraint, parentheses are required around the foreign key column name.
   - True
   - False

3) In a FOREIGN KEY constraint, data types of the foreign key and primary

key columns must be the same.

- True
- False

4) Adding a FOREIGN KEY constraint to a table only affects inserting new rows into the table.

- True
- False

CHALLENGE ACTIVITY | 4.4.1: Foreign key constraints.

544874.3500394.qx3zqy7

Start

Country

● TLD	CountryName	Area	IndependenceYear
.er	Eritrea	38996.31801	1991
.ng	Nigeria	351650.2629	1960
.mu	Mauritius	783.7873818	1968

Geography

● ISOCode2	Code	PopDensity	Continent
ER	.er	136.4231361	Africa
NG	.ng	557.0157644	Africa
MU	.mu	1614.344693	Africa

What is the PopDensity of Nigeria?

1 2 3

Check Next

Exploring further:

- MySQL FOREIGN KEY constraint

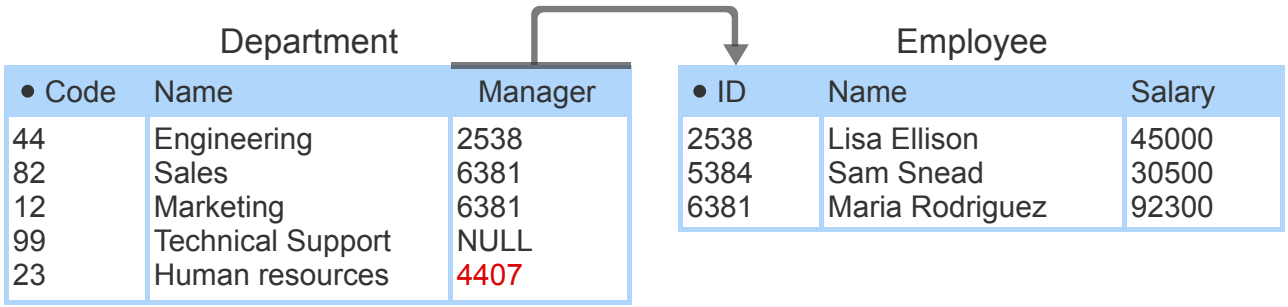
4.5 Referential integrity

Referential integrity rule

A ***fully NULL*** foreign key is a simple or composite foreign key in which all columns are NULL. ***Referential integrity*** is a relational rule that requires foreign key values are either fully NULL or match some primary key value.

In a relational database, foreign keys must obey referential integrity at all times. Occasionally, data entry errors or incomplete data result in referential integrity violations. Violations must be corrected before data is stored in the database.

PARTICIPATION ACTIVITY 4.5.1: Referential integrity violations.



Department

• Code	Name	Manager
44	Engineering	2538
82	Sales	6381
12	Marketing	6381
99	Technical Support	NULL
23	Human resources	4407

Employee

• ID	Name	Salary
2538	Lisa Ellison	45000
5384	Sam Snead	30500
6381	Maria Rodriguez	92300

HealthPlan

• PlanNumber	PlanName	EmployeeID	DependentNumber
323	Blue Shield	6381	1
552	Anthem	6381	2

926	Anthem	6381	3
801	UnitedHealthcare	6381	4
666	UnitedHealthcare	NULL	1

Family

• ID	• Number	Relationship	Name
2538	1	Spouse	Henry Ellison
2538	2	Son	Edward Ellison
6381	1	Spouse	Jose Rodriguez
6381	2	Daughter	Gina Rodriguez
6381	3	Daughter	Clara Rodriguez

Animation content:

Step 1: 4407 does not match any value in ID and violates referential integrity. The Department and Employee tables appear. Department has columns Code, Name, and Manager. Employee has columns ID, Name, and Salary. ID is the primary key. An arrow points from the Manager column to the ID column. Row five of column Manager is highlighted and contains the value 4407.

Step 2: (6381, 4) does not match any value in (ID, Number) and violates referential integrity. The HealthPlan and Family tables appear. Health Plan has columns PlanNumber, PlanName, EmployeeID, and DependentNumber. Family has columns ID, Number, Relationship, and Name. (ID, Number) is the primary key. An arrow points from (EmployeeID, DependentNumber) of the HealthPlan table to (ID, Number) of the Family table. The value (6381, 4) in (EmployeeID, DependentNumber) is highlighted.

Step 3: (NULL, 1) is partially NULL and violates referential integrity. The value (NULL, 1) in (EmployeeID, DependentNumber) is highlighted.

Animation captions:

1. 4407 does not match any value in ID and violates referential integrity.
2. (6381, 4) does not match any value in (ID, Number) and violates referential integrity.
3. (NULL, 1) is partially NULL and violates referential integrity.

PARTICIPATION ACTIVITY | 4.5.2: Referential integrity rules for simple primary keys.

Refer to the tables below.

Department → Employee

• Code	Name	Manager
44	Engineering	2538
82	Sales	3829
12	Marketing	6381
99	Technical Support	NULL

• ID	Name	Salary
2538	Lisa Ellison	45000
5384	Sam Snead	30500
6381	Maria Rodriguez	92300

1) In the Department table, which foreign key value violates referential integrity?

- 2538
- 3829
- NULL

2) Does the NULL in the Manager column violate referential integrity?

- Yes
- No

PARTICIPATION ACTIVITY 4.5.3: Referential integrity rules for composite foreign keys.

Refer to the tables below.

HealthPlan

• PlanNumber	PlanName	EmployeeID	DependentNumber
323	Blue Shield	6381	1
552	Anthem	6381	2
926	Anthem	6381	3
801	UnitedHealthCare	6381	4
666	UnitedHealthCare	6381	NULL
744	Blue Shield	NULL	NULL

Family

• ID	• Number	Relationship	Name
2538	1	Spouse	Henry Ellison
2538	2	Son	Edward Ellison
6381	1	Spouse	Jose Rodriguez
6381	2	Daughter	Gina Rodriguez
6381	3	Daughter	Clara Rodriguez

1) In the HealthPlan table, which foreign key value violates referential integrity?

- (NULL, NULL) only
- (6381, NULL) only
- (6381, 4) only
- Both (6381, NULL) and (6381, 4)

2) In the HealthPlan table, which foreign key value is fully NULL?

- (6381, NULL)
- (NULL, NULL)
- No foreign key values are fully NULL

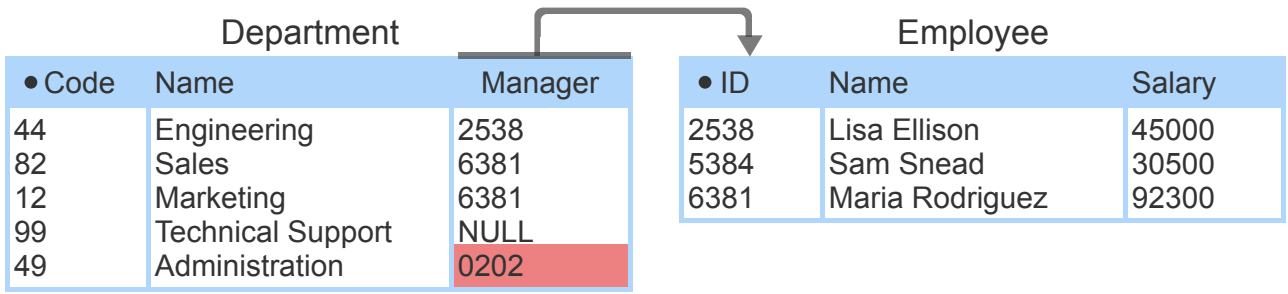
Referential integrity violations

Referential integrity can be violated in four ways:

1. A primary key is updated.
2. A foreign key is updated.
3. A row containing a primary key is deleted.
4. A row containing a foreign key is inserted.

Only these four operations can violate referential integrity. Primary key inserts and foreign key deletes never violate referential integrity.

PARTICIPATION ACTIVITY 4.5.4: Four ways to violate referential integrity.



Animation content:

Step 1: Updating the Employee primary key to 8888 violates referential integrity because the foreign key 2538 no longer exists in Employee. There are two tables named Department and Employee. Department has columns Code, Name, and Manager. Employee has columns ID, Name, and Salary. ID is the primary key. An arrow points from Manager to ID. Row one of columns Manager and ID are highlighted and both contain the value 2538. The value in row one of column ID changes from 2538 to 8888.

Step 2: Updating the foreign key to 3333 violates referential integrity because 3333 does not match a primary key value. Row two of column Manager and row three of column ID contain the value 6381 and are highlighted. The value in row two of column Manager changes from 6381 to 3333.

Step 3: Deleting Employee primary key 6381 violates referential integrity because the foreign key 6381 no longer exists in Employee. Row three of the Employee table is highlighted:
6381, Maria Rodriguez, and 92300
A line strikes through this row. Rows two and three of column Manager are highlighted and both contain the values 6381.

Step 4: Inserting foreign key 0202 violates referential integrity because 0202 does not match a primary key value.A new row is added to the Department table:
49, Administration, 0202
The value 0202 in this row is highlighted.

Animation captions:

1. Updating the Employee primary key to 8888 violates referential integrity because the foreign key 2538 no longer exists in Employee.
2. Updating the foreign key to 3333 violates referential integrity because 3333 does not match a primary key value.
3. Deleting Employee primary key 6381 violates referential integrity because the foreign key 6381 no longer exists in Employee.
4. Inserting foreign key 0202 violates referential integrity because 0202 does not match a primary key value.

PARTICIPATION ACTIVITY 4.5.5: Referential integrity violations.

Refer to the tables in the above animation. Match the violation type to the database change.

If unable to drag and drop, refresh the page.

Update a primary key **Delete a primary key** **Update a foreign key**

Insert a foreign key

	Change Maria Rodriguez's ID to 9925.
	Change the Technical Support department manager to 8001.
	Remove Lisa Ellison from the Employee table.
	Add Human Resources to the Department table with manager 1420.

Reset

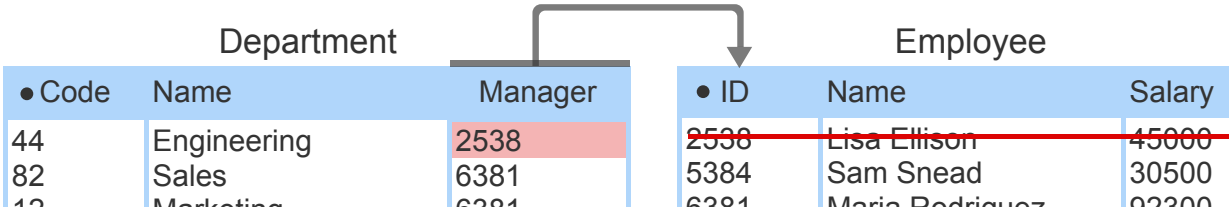
Referential integrity actions

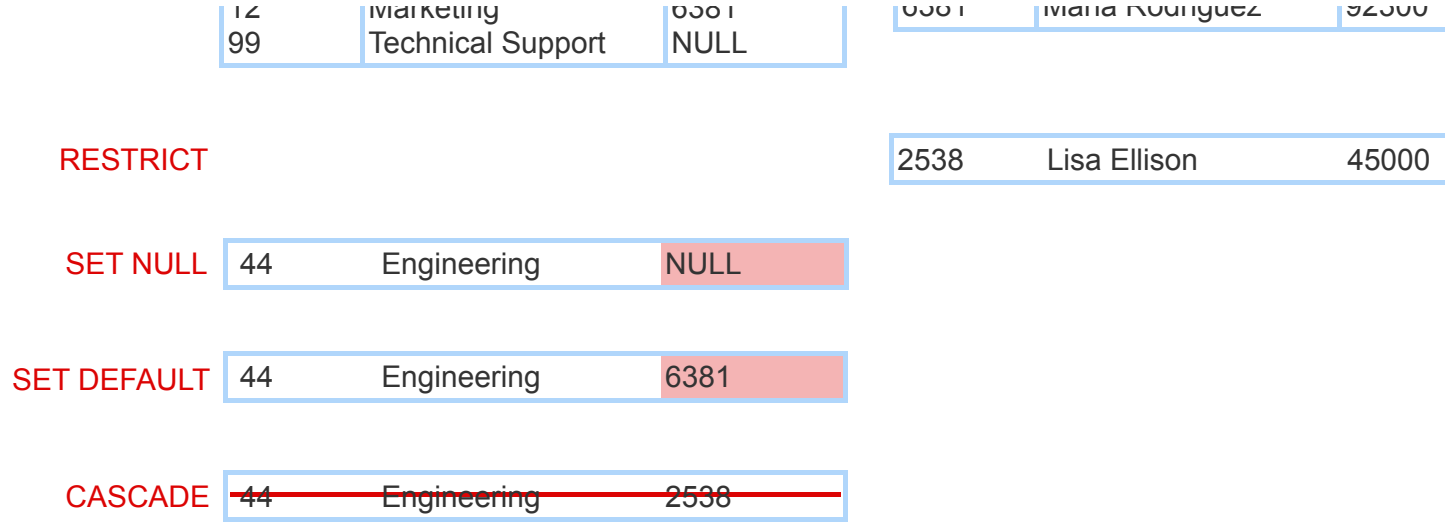
An insert, update, or delete that violates referential integrity can be corrected manually. However, manual corrections are time-consuming and error-prone. Instead, databases automatically correct referential integrity violations with any of four actions, specified as SQL constraints:

- ***RESTRICT*** rejects an insert, update, or delete that violates referential integrity.
- ***SET NULL*** sets invalid foreign keys to NULL.
- ***SET DEFAULT*** sets invalid foreign keys to the foreign key default value.
- ***CASCADE*** propagates primary key changes to foreign keys.

CASCADE behaves differently for primary key updates and deletes. If a primary key is deleted, rows containing matching foreign keys are deleted. If a primary key is updated, matching foreign keys are updated to the same value.

PARTICIPATION ACTIVITY 4.5.6: Referential integrity actions on primary key delete.





Animation content:

Static figure:
The Department table has columns Code, Name, and Manager. Code is the primary key. Department has four rows. The first row is:
44, Engineering, 2538

The Employee table has columns ID, Name, and Salary. ID is the primary key. Employee has three rows. The first row is:
2538, Lisa Ellison, 45000

An arrow points from the Manager column of Department to the ID column of Employee.

Step 1: The row containing primary key 2538 is deleted. A line strikes through the Employee row:
44, Engineering, 2538

Step 2: RESTRICT rejects the delete, since employee 2538 manages Engineering. A row labeled RESTRICT appears below the Employee table:
2538, Lisa Ellison, 45000
A line strikes through this row.

Step 3: SET NULL sets matching foreign keys to NULL. A row labeled SET NULL appears below the Department table:
44, Engineering, NULL
The NULL value is highlighted.

Step 4: SET DEFAULT sets matching foreign keys to the foreign key default value, 6381. A row labeled SET DEFAULT appears below the Department table:

labeled SET DEFAULT appears below the Department table.
44, Engineering, 6381
The value 6381 is highlighted.

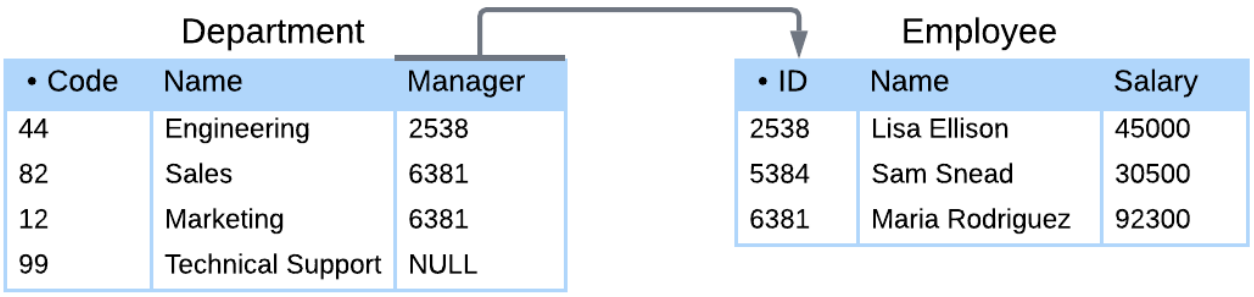
Step 5: CASCADE deletes all rows with matching foreign key values. A line strikes through the Engineering row in the Department table. A row labeled CASCADE appears below the Department table:
44, Engineering, 2538
A line strikes through this row.

Animation captions:

1. The row containing primary key 2538 is deleted.
2. RESTRICT rejects the delete, since employee 2538 manages Engineering.
3. SET NULL sets matching foreign keys to NULL.
4. SET DEFAULT sets matching foreign keys to the foreign key default value, 6381.
5. CASCADE deletes all rows with matching foreign key values.

PARTICIPATION ACTIVITY 4.5.7: Referential integrity actions.

Refer to the tables below. What are the results of the following actions?



1) RESTRICT, when the row containing Maria Rodriguez is deleted.

- The Sales and Marketing managers are set to NULL.
- The Sales and Marketing departments are deleted.
- The delete is rejected.

2) SET NULL, when Lisa Ellison's ID is

changed to 1001.

- The Engineering manager is set to NULL.
- The Engineering manager is set to 1001.
- The change is rejected.

3) SET DEFAULT, when Lisa Ellison's ID is changed to 1001.

- The Engineering manager is set to NULL.
- The Engineering manager is set to the Manager default value.
- The change is rejected.

4) CASCADE, when Maria Rodriguez' ID is changed to 2022.

- The Sales and Marketing managers are set to NULL.
- The Sales and Marketing managers are set to 2022.
- The change is rejected.

5) CASCADE, when Maria Rodriguez is deleted.

- The Sales and Marketing managers are set to NULL.
- The Sales and Marketing departments are deleted.
- The delete is rejected.

ON UPDATE and ON DELETE clauses

For foreign key inserts and updates, MySQL supports only RESTRICT. Foreign key inserts and updates that violate referential integrity are automatically rejected.

For primary key updates and deletes, MySQL supports all four actions. Actions are specified in the optional ***ON UPDATE*** and ***ON DELETE*** clauses of the FOREIGN KEY constraint. ON UPDATE and ON

DELETE are followed by either RESTRICT, SET NULL, SET DEFAULT, or CASCADE.

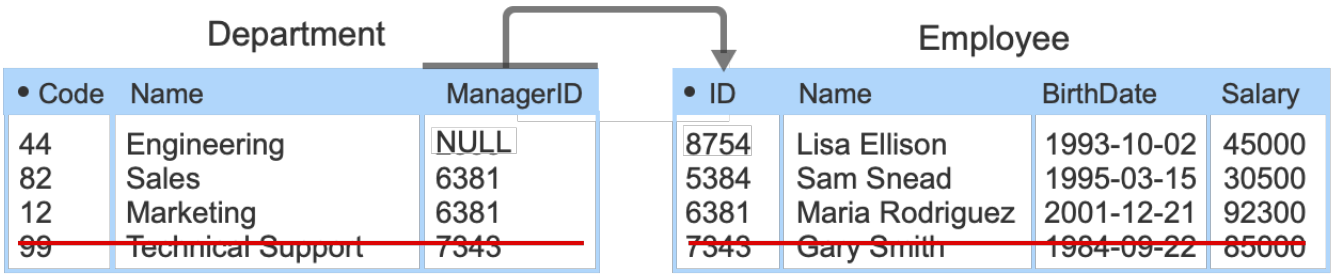
ON UPDATE and ON DELETE determine what happens to the foreign key when the referenced primary key is updated or deleted. When several foreign keys refer to the same primary key, different actions can be specified for each foreign key.

MySQL has several limitations on primary key updates and deletes:

- RESTRICT is applied when the ON UPDATE or ON DELETE clause is omitted.
- SET NULL cannot be used when a foreign key is not allowed NULL values.
- SET DEFAULT is not supported in some MySQL configurations.

ON UPDATE and ON DELETE are standard SQL. The clauses are supported by most relational databases, but details and limitations vary.

PARTICIPATION ACTIVITY 4.5.8: Foreign key constraints with ON UPDATE and ON DELETE clauses.




```
CREATE TABLE Department (
   Code      TINYINT UNSIGNED,
   Name      VARCHAR(20),
   ManagerID SMALLINT UNSIGNED,
   PRIMARY KEY (Code),
   FOREIGN KEY (ManagerID) REFERENCES Employee(ID)
      ON DELETE CASCADE
      ON UPDATE SET NULL
);
```

Animation content:

Static figure:
The Department table has columns Code, Name, and ManagerID. Code is the primary key.
Department has four rows:
44, Engineering, 2538

82, Sales, 6381
12, Marketing, 6381
99, Technical Support, 7343

The Employee table has columns ID, Name, BirthDate, and Salary. ID is the primary key. Employee has four rows:
2538, LIsa Ellison, 1993-10-02, 45000
5384, Sam Snead, 1995-03-15, 30500
6381, Maria Rodriguez, 2001-12-21, 92300
7343, Gary Smith, 1984-09-22, 85000

An arrow points from ManagerID to ID.

An SQL statement appears:
Begin SQL code:

```
CREATE TABLE Department (
  Code TINYINT UNSIGNED,
  Name VARCHAR(20),
  ManagerID SMALLINT UNSIGNED,
  PRIMARY KEY (Code),
  FOREIGN KEY (ManagerID) REFERENCES Employee(ID)
    ON DELETE CASCADE
    ON UPDATE SET NULL
);
```

End SQL code.

Step 1: ManagerID is a foreign key that references the Employee ID column. The FOREIGN KEY clause is highlighted. The arrow from ManagerID to ID appears.

Step 2: ON DELETE CASCADE causes the database to delete the row with ManagerID 7343 when the employee with ID 7343 is deleted. The ON DELETE clause is highlighted. A line strikes out the fourth row of Employee:
7343, Gary Smith, 1984-09-22, 85000
A line strikes out the fourth row of Department:
99, Technical Support, 7343

Step 3: ON UPDATE SET NULL causes the database to set ManagerID 2538 to NULL when the Employee ID 2538 is changed to 8754. The ON UPDATE clause is highlighted. In the ID column of Employee, the value 2538 changes to 8754. In the ManagerID column of Department, the value 2538 changes to NULL.

Animation captions:

1. ManagerID is a foreign key that references the Employee ID column.
2. ON DELETE CASCADE causes the database to delete the row with ManagerID 7343 when the employee with ID 7343 is deleted.
3. ON UPDATE SET NULL causes the database to set ManagerID 2538 to NULL when the Employee ID 2538 is changed to 8754.

PARTICIPATION ACTIVITY 4.5.9: ON UPDATE and ON DELETE clauses.

Refer to the table definition and data below.

```
CREATE TABLE Department (
   Code TINYINT UNSIGNED,
   Name VARCHAR(20),
   ManagerID SMALLINT UNSIGNED,
   PRIMARY KEY (Code),
   FOREIGN KEY (ManagerID) REFERENCES Employee(ID)
      ON DELETE SET NULL
      ON UPDATE CASCADE
);
```

Department

• Code	Name	Manager
44	Engineering	2538
82	Sales	6381
12	Marketing	6381
99	Technical Support	NULL
49	Administration	7343

Employee

• ID	Name	Salary
2538	Lisa Ellison	45000
5384	Sam Snead	30500
6381	Maria Rodriguez	92300
7343	Gary Smith	85000

What is the result of each operation?

1) Delete Lisa Ellison.

- Lisa Ellison is deleted.
- Lisa Ellison is deleted, and the Engineering ManagerID is set to NULL.
- The delete is rejected.

2) Update Lisa Ellison's ID to 1000.

- Lisa Ellison's ID is set to 1000.

- Lisa Ellison's ID and the Engineering ManagerID are set to 1000.
- The update is rejected.

3) Update the Engineering ManagerID to 9999.

- The Engineering ManagerID is set to 9999.
- Lisa Ellison's ID and the Engineering ManagerID are set to 9999.
- The update is rejected.

4) Delete Engineering.

- Engineering is deleted.
- Engineering is deleted, and Lisa Ellison's ID is set to NULL.
- The delete is rejected.

CHALLENGE ACTIVITY | 4.5.1: Referential integrity.

544874.3500394.qx3zqy7

Start

Country

• Code	Name	PopDensity	IndepYear
450	Madagascar	116.9116894	1960
380	Italy	532.0311417	NULL
686	Senegal	213.2789898	1960

Geography

• ISOCode3	ID	Over65PopPct	Continent
MDG	450	0.029	Africa
ITA	380	0.23	Europe
SEN	686	0.03	Africa

With RESTRICT referential integrity, what happens if the row containing Madagascar is del

table?

Select

1 2 3 4 5

Check Next