What is a Database?

A database is defined as a collection of data organized especially for rapid search and retrieval.

1

Types of Databases

- Flat files
- NoSQL databases
- Data Warehouses
- Relational Databases

2

Flat Files

- Spreadsheet or text file used for storing data.
- Typically large un-normalized tables with many columns.
- Can be easy to view data.
- Can be difficult to update and maintain.
- Prone to data redundancy (i.e. unnecessary duplication).
- Prone to data inconsistencies and inaccuracies which are likely to occur over time.

3

NoSQL Databases

- Sometimes referred to as “Not Only SQL databases”.
- Data storage and retrieval is modeled in means other than tabular relations.
- Commonly used in Big Data and Real-Time Applications.
- Many different types of NoSQL Databases including Graph and Key-value databases.

4

Data Warehouses

- A special type of database which are optimized for reporting and data analysis purposes.
- Primarily designed for analyzing historical data.
- Historical data typically sourced from relational databases.
- Separate from an organization's operational database.

Relational Databases

- Relational Database Management Systems (RDMS) provide a standard approach for storing and querying data.
- Data is queried using Structured Query Language (SQL).
- Standard approach for enforcing data integrity.
- Minimization of data redundancy – 'store data only once'.

Logical Design vs. Physical Design

Logical Design:

Logical Design for relational databases involves arranging data into tables and defining attributes and relationships between tables.

Physical Design:

Physical Design is the actual process of creating the tables, constraints, relationships, etc. in SQL.

7

Logical Design vs. Physical Design

It's generally recommended to do a complete logical design first…

then build the physical design from the logical.

8

Common Table Terminology

9

This table has 4 columns and 5 rows

Social Security Number	Employee ID	First Name	Last Name
751-03-1503	1	Andrew	Rivers
039-54-4183	2	Vera	Hill
520-05-0425	3	Ben	Jones
662-10-5060	4	Ben	Jones
257-34-8033	5	Chelsea	Walsh

10

Column names: Also called column headers, field names, or attributes

Social Security Number	Employee ID	First Name	Last Name
751-03-1503	1	Andrew	Rivers
039-54-4183	2	Vera	Hill
520-05-0425	3	Ben	Jones
662-10-5060	4	Ben	Jones
257-34-8033	5	Chelsea	Walsh

11

5 rows total in this table. Rows are also called records

Social Security Number	Employee ID	First Name	Last Name
751-03-1503	1	Andrew	Rivers
039-54-4183	2	Vera	Hill
520-05-0425	3	Ben	Jones
662-10-5060	4	Ben	Jones
257-34-8033	5	Chelsea	Walsh

12

Introduction to Data Integrity

- Data Integrity refers to the accuracy and consistency of data.
- In Relational Databases we can enforce data integrity by adding **constraints** to tables.
- The most important types of constraints are:
  - Primary key constraints
  - Foreign key constraints
  - Not null constraints
- Other types of constraints include:
  - Unique constraints
  - Check constraints

13

Introduction to NULL values

Definition:

A value of NULL is an unknown value. NULL values are not the same as zeros, empty values or character strings. A NULL value is not equal to anything:

NULL is not equal to NULL

NULL != NULL

(4 x 5) + NULL = NULL

Operator	Definition
=	Equal
<	Less than
>	Greater than
!=	Not equal
< >	Not equal

14

Handling Null Values

Mathematical errors with nulls can easily go undetected in queries and views:

Emp ID	Emp Last Name	Salary	Bonus	Total Remuneration
1	Jones	60,000	800	60800
2	Richardson	65,000	<null>	<null> ✗

```
SELECT
    EMP_ID,
    EMP_LAST_NAME,
    SALARY,
    BONUS,
    SALARY + BONUS AS TOTAL_REMUNERATION
FROM EMP1;
```

15

The NOT NULL Constraint

Definition:

A not null constraint on a column does not allow a null value to be inserted into that column.

If you want a column to always contain a non-null value then use a not null constraint.

16

Primary Key Constraints

- A primary key constraint is placed on a column or set of columns. This constraint ensures that the values in the primary key column(s) are both **unique** and **not null**.
- A single table might have many potential (candidate) keys but only one candidate key is selected to have a primary key constraint placed on it.
- A candidate key is essentially a column or combination of columns which has unique values.

17

Composite Primary Keys

- Refers to where more than one column is used in a table's primary key.
- A composite primary key forces the combination of columns included in the key to be both unique and not null.
- Composite primary keys can be cumbersome (slow) to use in joins.
- Associative tables (used to handle many-to-many) relationships are where you will typically implement composite primary key.

18

Natural Keys

Natural Key:
A column or combination of columns that uniquely identifies each row in a table where the key columns are made up of real-world data.

e.g. Park Name

Park Name	Park Type
Yellowstone	National
Congaree	National
Baxter	State

19

Multi-Column Natural Key

Supplier Name	Product Name	Order Datetime	Quantity
Davies Autoparts	V2 spark plug	2018-11-24 13:24:11	4
ABC Ltd	A11 windscreen	2018-11-24 11:59:01	4
ABC Ltd	Standard Tyre	**2018-11-24 10:23:06**	5
ABC Ltd	Standard Tyre	**2018-11-23 11:45:07**	6

{SupplierName, ProductName, OrderDatetime}
is a natural candidate key in this table.

20

Surrogate Keys

Surrogate Key:

A column that uniquely identifies each row in a table and is typically system generated.

e.g. employee ID

Question:

Is a book's ISBN a natural key or a surrogate key?

ISBN's were created in 1970 as a surrogate key for books. Books are naturally identified by attributes such as title, author, publication date etc.

21

Main Benefit of Surrogate Keys

- Surrogate (system-generated) keys are guaranteed to never change.

Country ID	Country Name
1	Burma
2	Rhodesia
3	Upper Volta

22

Main Benefit of Surrogate Keys

- Surrogate (system-generated) keys are guaranteed to never change.

Country ID	Country Name
1	Myanmar
2	Zimbabwe
3	Burkina Faso

23

Primary Key (PK) Recommendations

- It is good practice for every table to have a PK constraint.
- Never select a column whose values might change in the future as the PK
  - E.g. Phone number or email address are bad choices for a PK.
- Often a good idea to use a surrogate key as the PK
  - E.g. Auto-incrementing number
- If you use a surrogate key as the PK then identify any natural keys in the table and put unique constraints or unique indexes on these.

24

Primary key constraints vs. Unique constraints

Primary Key Constraint:	Unique Constraint:
▪ A single table can only have **one** primary key constraint. ▪ A primary key ensures that values are unique and NOT NULL. ▪ A primary key automatically creates a unique index.	▪ A single table can have **multiple** unique constraints. ▪ A unique constraint ensures values are unique, however, nulls are allowed. ▪ A unique constraint automatically creates a unique index.

25

Indexes

An index is a data structure which *can* speed up searches on a table if properly implemented.

Basic syntax example:

```
CREATE INDEX index_name ON table_name;
```

Note:

When a table is dropped, then all the associated indexes and constraints are also dropped.

26

Unique Indexes

- A unique index ensures that a column or columns will contain unique values.
- When either a unique constraint or primary key constraint is created then a corresponding unique index is also created.
- By default this unique index has the same name as the constraint.

```
CREATE UNIQUE INDEX index_name ON table_name (column_name)
```


Types of Relationships between Entities

One-to-Many:

Occurs when an instance of **Entity A** can potentially be associated with many instances of **Entity B.**

e.g. A single car is made of many parts.

Many-to-Many:

Occurs when an instance of **Entity A** can potentially be associated with many instances of **Entity B.** In addition, a single instance of **Entity B** can potentially be associated with many instances of **Entity A.**

e.g. A single guest can book many hotel rooms and a single room can have many guests.

One-to-One:

Occurs when an instance of **Entity A** is associated to either zero or one instance of **Entity B.**

e.g. A single country can only have one capital city.

Identifying Relationships Between Entities

In order to determine the relationship between a pair of entities you need to ask two questions:

Can a single instance of **Entity A** be associated with one or many instances of **Entity B?**

Then ask the question in reverse:

Can a single instance of **Entity B** be associated with one or many instances of **Entity A?**

29

Identifying Relationships Between Entities

Entity A: Employees **Entity B**: Departments

Question 1:
Can an Employee be associated with one or many Departments?
A single Employee can only belong to ***one*** *Department.*

Question 2:
Can a single Department be associated with one or many Employees?
A single department can have ***many*** *Employees.*

Conclusion:
There is a **one-to-many** relationship between Employees and Departments.

30

Identifying Relationships Between Entities

Entity A: Doctors ***Entity B:*** Patients

Question 1:
Can a Doctor be associated with one or many Patients?
*A single Doctor can see **many** Patients.*

Question 2:
Can a single Patient be associated with one or many Doctors?
*A single Patient can see **many** Doctors.*

Conclusion:
There is a **many-to-many** relationship between Doctors and Patients.

31

One-to-Many Relationships

32

(Parent Table)

Departments Table		
Phone	**Dept Name**	**Dept ID**
843-124-09	Accounting	101
843-189-10	Marketing	102
843-254-11	Sales	103
843-128-04	IT	104

Primary Key (PK)

(Child Table)

Employees Table			
Department ID	**Employee ID**	**First Name**	**Last Name**
101	1	Andrew	Rivers
101	2	Vera	Hill
102	3	Ben	Jones
103	4	Ben	Jones
103	5	Chelsea	Walsh

Foreign key (FK) *Primary Key (PK)*

(PK) = Primary Key
(FK) = Foreign Key

```
…CONSTRAINT EMPLOYEES_DEPT_ID_FK FOREIGN KEY (DEPARTMENT_ID)
REFERENCES DEPARTMENTS (DEPT_ID));
```

33

One-to-Many Relationships

A one-to-many relationship occurs between a pair of tables (i.e. entities) when a single record in the parent table can potentially be related to many records in the child table.

In a one-to-many relationship a single record in the child table can be related to *only one* record in the parent table.

A foreign key constraint on a column ensures that the values in that column must also appear in a referenced column which is typically the primary key in the parent table.

34

(Parent Table)

Departments Table		
Phone	**Dept Name**	**Dept ID**
843-124-09	Accounting	101
843-189-10	Marketing	102
843-254-11	Sales	103
843-128-04	IT	104

Primary Key (PK)

(Child Table)

Employees Table			
Department ID	**Employee ID**	**First Name**	**Last Name**
101	1	Andrew	Rivers
101	2	Vera	Hill
102	3	Ben	Jones
103	4	Ben	Jones
103	5	Chelsea	Walsh

Foreign key(FK) ***Primary Key (PK)***

(PK) = Primary Key
(FK) = Foreign Key

```
…CONSTRAINT EMPLOYEES_DEPT_ID_FK FOREIGN KEY (DEPARTMENT_ID)
REFERENCES DEPARTMENTS (DEPT_ID));
```

35

(Parent Table)

Departments Table		
Phone	**Dept Name**	**Dept ID**
843-124-09	Accounting	101
843-189-10	Marketing	102
843-254-11	Sales	103
843-128-04	IT	104

Primary Key (PK)

(Child Table)

Employees Table			
Department ID	**Employee ID**	**First Name**	**Last Name**
101	1	Andrew	Rivers
101	2	Vera	Hill
102	3	Ben	Jones
103	4	Ben	Jones
103	5	Chelsea	Walsh

Foreign key (FK) ***Primary Key (PK)***

(PK) = Primary Key
(FK) = Foreign Key

```
…CONSTRAINT EMPLOYEES_DEPT_ID_FK FOREIGN KEY (DEPARTMENT_ID)
REFERENCES DEPARTMENTS (DEPT_ID));
```

36

(Parent Table)

Departments Table		
Phone	**Dept Name**	**Dept ID**
843-124-09	Accounting	101
843-189-10	Marketing	102
843-254-11	Sales	103
843-128-04	IT	104

Primary Key (PK)

(Child Table)

Employees Table			
Department ID	**Employee ID**	**First Name**	**Last Name**
101	1	Andrew	Rivers
101	2	Vera	Hill
102	3	Ben	Jones
103	4	Ben	Jones
103	5	Chelsea	Walsh

Foreign key(FK) ***Primary Key (PK)***

(PK) = Primary Key
(FK) = Foreign Key

```
…CONSTRAINT EMPLOYEES_DEPT_ID_FK FOREIGN KEY (DEPARTMENT_ID)
REFERENCES DEPARTMENTS (DEPT_ID));
```

37

(Parent Table)

Departments Table		
Phone	**Dept Name**	**Dept ID**
843-124-09	Accounting	101
843-189-10	Marketing	102
843-254-11	Sales	103
843-128-04	IT	104

Primary Key (PK)

(Child Table)

Employees Table			
Department ID	**Employee ID**	**First Name**	**Last Name**
101	1	Andrew	Rivers
101	2	Vera	Hill
102	3	Ben	Jones
103	4	Ben	Jones
103	5	Chelsea	Walsh

Foreign key(FK) ***Primary Key (PK)***

(PK) = Primary Key
(FK) = Foreign Key

```
…CONSTRAINT EMPLOYEES_DEPT_ID_FK FOREIGN KEY (DEPARTMENT_ID)
REFERENCES DEPARTMENTS (DEPT_ID));
```

38

(Parent Table)

Departments Table		
Phone	**Dept Name**	**Dept ID**
843-124-09	Accounting	101
843-189-10	Marketing	102
843-254-11	Sales	103
843-128-04	IT	104

Primary Key (PK)

(Child Table)

Employees Table			
Department ID	**Employee ID**	**First Name**	**Last Name**
101	1	Andrew	Rivers
101	2	Vera	Hill
102	3	Ben	Jones
103	4	Ben	Jones
103	5	Chelsea	Walsh

Foreign key(FK) *Primary Key (PK)*

(PK) = Primary Key
(FK) = Foreign Key

```
…CONSTRAINT EMPLOYEES_DEPT_ID_FK FOREIGN KEY (DEPARTMENT_ID)
REFERENCES DEPARTMENTS (DEPT_ID));
```

39

Departments Table		
Phone	**Dept Name**	**Dept ID**
843-124-09	Accounting	101
843-189-10	Marketing	102
843-254-11	Sales	103
843-128-04	IT	104

Employees Table			
Department ID	**Employee ID**	**First Name**	**Last Name**
101	1	Andrew	Rivers
101	2	Vera	Hill
102	3	Ben	Jones
103	4	Ben	Jones
103	5	Chelsea	Walsh

Employees Dept View					
Employee ID	**First Name**	**Last Name**	**Dept ID**	**Dept Name**	**Dept Phone Contact**
1	Andre w	Rivers	101	Accounting	843-124-09
2	Vera	Hill	101	Accounting	843-124-09
3	Ben	Jones	102	Marketing	843-189-10
4	Ben	Jones	103	Sales	843-254-11
5	Chelse a	Walsh	103	Sales	843-254-11

```
CREATE VIEW EMPLOYEES_DEPT AS
SELECT
  E.EMP_ID,
  E.FIRST_NAME,
  E.LAST_NAME,
  E.DEPARTMENT_ID,
  D.DEPT_NAME,
  D.PHONE
FROM EMPLOYEES E INNER JOIN DEPARTMENTS D
ON E.DEPARTMENT_ID = D.DEPT_ID;
```

40

Foreign Key Constraint Summary

- Foreign keys are used to enforce referential integrity between tables.
- In a one-to-many relationship the foreign key constraint is placed on the many-side (child) table.
- The values of a foreign key column uniquely identify a row of another table *or the same table*.
- The foreign key *usually* refers to the primary key in another table.
- To represent a one-to-many relationship we take the primary key from the parent table and add this column as a foreign key in the child table.

41

One-to-One Relationships

Occurs when an instance of **Entity A** is associated to either zero or one instance of **Entity B.**

Typically the attributes are put in the same table where a one-to-one relationship exists. An exception to this is for security purposes.

A one-to-one relationship can be implemented by putting both a foreign key and a primary key (or unique index) on the same column on one of the tables. This table then becomes the child table.

42

One-to-One Relationship Example

Employees Table		
Emp ID	**First Name**	**Last Name**
1	Andrew	Rivers
2	Vera	Bowman
3	Adam	Jones
4	Adam	Jones
5	Chelsea	Walsh

Primary Key (PK)

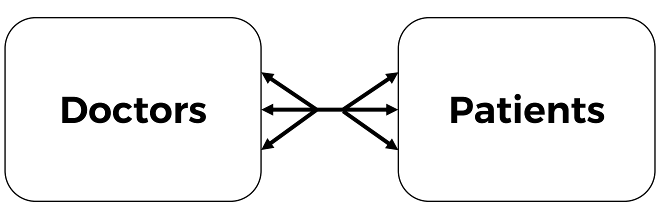
Compensation Table		
Emp ID	**Salary**	**Health Insurance Plan**
1	70000	Medical Basic
2	70000	Full Medical
3	65000	Medical Basic
4	80000	Full Medical
5	650000	Medical Basic

(PK) / (FK)

43

Many-to-Many Relationships

A many-to-many relationship exists between two entities when an instance of **Entity A** can potentially be associated with many instances of **Entity B**. In addition, a single instance of **Entity B** can potentially be associated with many instances of **Entity A**.



Each doctor sees many patients and each patient may see many doctors.

44

Many-to-Many Relationships

Many-to-many relationships are often incorrectly modelled by duplicating data in one of the tables.

The correct way is to create an associative table. This is also called a junction table or a linking table.

45

Many-to-Many Relationship Example

Parent Table

Doctors	
DocID	**Doctor Surname**
1001	Jackson
1002	Anderson
1003	Robbins

(PK)

Child Table

DoctorPatients	
DocID	**PatID**
1001	1
1001	2
1002	3
1002	2
1003	2
1003	3

(FK) ***(FK)***

$\underbrace{\hspace{6em}}$

Composite Primary key

Parent Table

Patients	
PatID	**Patient Surname**
1	Perkins
2	Reeves
3	Shapiro

(PK)

46

Self-Referencing Relationships

Employee ID	Emp First Name	Manager ID	Position
1	Judy	<null>	CEO
2	James	1	Sales Manager
3	Amanda	1	Accountant
4	Bob	2	Sales Rep
5	Henry	7 ❌	Sales Rep

Henry's manager is given as employee ID 7.
However, there is no Employee ID 7 in this table.

47

Data Normalization

- First proposed by E.F. (Ted) Codd in 1970 and has become the cornerstone of relation database design.
- A process of organizing attributes and tables to minimise data redundancy i.e. *unnecessary duplication of data.*
- Typically involves breaking larger tables into smaller (less redundant) tables and defining relationships between them.

48

Why Normalize Data?

- Eliminates the potential for **Update**, **Insert**, and **Deletion** Anomalies to occur.
- Reduces the need for restructuring data as new types of data are introduced to the database.
- Makes the data more informative to users.

49

Normal Forms

There are different levels of data normalization which are referred to as ***normal forms***

- At least five normal forms have been described.
- Normal forms build on each other:
  - If a table is in 2NF then it is also in 1NF.
  - If a table is in 3NF then it is also in 1NF and 2NF.

50

Functional Dependencies

A functional dependency is a type of relationship that occurs when one attribute uniquely determines another attribute

If X functionally determines Y (i.e. $X \rightarrow Y$) then every value of X determines ***exactly one*** value of Y.

Example:
If I know the value of attribute **X**, then I know the unique value for attribute **Y.**

Attribute **X** functionally determines Attribute **Y** **($X \rightarrow Y$)**
which is the same as saying…
Attribute **Y** is functionally dependent on Attribute **X**

51

Functional Dependency Example

If I know a student's ID number then I can tell you with certainty that student's first name.

In other words:

StudentID functionally determines student First Name:

StudentID → FirstName

FirstName does not functionally determine **StudentID**

52

Trivial Functional Dependencies

Emp ID	Dept ID	Emp First Name	Position
1	100	Judy	CEO
2	100	James	Sales Manager
3	101	Amanda	Accountant
4	102	Bob	Sales Rep
5	102	Henry	Sales Rep

DeptID → DeptID

{EmpID, DeptID} → DeptID

A functional dependency is trivial if the right-hand side of the functional dependency (the dependent) is a subset of the left-hand side (the determinant).

53

Candidate Keys

A candidate key is either a single attribute with unique values or a set of attributes that have unique combinations of values

and…

A set of attributes can only be candidate key if it is *irreducible* i.e. it has no unique subset of attributes.

A candidate key functionally determines all the other columns in a table:

StudentID → FirstName, LastName, DOB

54

Candidate Keys

Student ID	First Name	Last Name	Date of Birth	SSN
1001	Harold	Wilson	19910806	142-33-8975
1002	John	Chang	19910806	132-38-8354
1003	Amy	Spears	19920731	142-43-3375

StudentID functionally determines all the other fields.

First Name currently has unique values. However, functional dependencies must always hold true. In this case, **First Name** does not functionally determine all other fields as in the future we might get more than one student with the same first name.

55

Candidate Keys

Student ID	First Name	Last Name	Date of Birth	SSN
1001	Harold	Wilson	19910806	142-33-8975
1002	John	Chang	19910806	132-38-8354
1003	Amy	Spears	19920731	142-43-3375

(StudentID, FirstName)

Recap:
A set of attributes can only be candidate key if has unique values and if it is has no unique subset of attributes. In other words if it cannot be reduced any further.

56

Candidate Keys

Student ID	First Name	Last Name	Date of Birth	SSN
1001	Harold	Wilson	19910806	142-33-8975
1002	John	Chang	19910806	132-38-8354
1003	Amy	Spears	19920731	142-43-3375

(FirstName,
FirstName)

57

Candidate Keys

Student ID	First Name	Last Name	Date of Birth	SSN
1001	Harold	Wilson	19910806	142-33-8975
1002	John	Chang	19910806	132-38-8354
1003	Amy	Spears	19920731	142-43-3375

58

Candidate Keys vs. Super Keys

Candidate Key:

A candidate key is an attribute or set of attributes that has the properties of both *uniqueness* and *irreducibility*.

Super Key:

A super key is a set of attributes that have the property of *uniqueness* but are not necessarily *irreducibility.*

In other words, all candidate keys are also super keys. But most super keys are not candidate keys.

59

Candidate Keys vs. Super Keys

Super Key: {SupplierName, ProductName, OrderDate, Quantity}

Supplier Name	ProductName	Order DateTime	Quantity
Davies Autoparts	V2 spark plug	2018-11-24 13:24:11	4
ABC Ltd	A11 windscreen	2018-11-24 11:59:01	4
ABC Ltd	Standard Tyre	2018-11-24 10:23:06	5
ABC Ltd	Standard Tyre	2018-11-23 11:45:07	6

Candidate Key: {SupplierName, ProductName, OrderDate}

60

Key and non-key Attributes

- Key attributes are simply attributes used in at least one candidate key
- Non-key attributes are any attributes that do not occur as part of any candidate key.

Note:

Key and non-key attributes are also referred to as prime and non-prime attributes.

61

Key and Non-key Attributes

Candidate Key: {SupplierName, ProductName, OrderDatetime}

Supplier Name Key attribute (Prime)	**ProductName** Key attribute (Prime)	**OrderDatetime** Key attribute (Prime)	**Quantity** Non- key attribute (Non-prime)
Davies Autoparts	V2 spark plug	2018-11-24	4
ABC Ltd	A11 windscreen	2018-11-24	4
ABC Ltd	Standard Tyre	2018-11-24	5
ABC Ltd	Standard Tyre	2018-11-23	6

62

First Normal Form (1NF)

63

1st Normal Form (1NF)

A table is in 1NF if:

The values for each attribute are of the same type and there are no duplicate rows and every attribute has rows with single (i.e. atomic) values.

Note:

Technically, nulls violate 1NF. This is because nulls are **not** true values and therefore cannot be of the appropriate type for an attribute.

64

Employee ID	First Name	Surname	Email
101	Judy	Robinson	judy.robinson@techlogic.net
101	Judy	Robinson	judy.robinson@techlogic.net
102	James	Lewis	(458) 904-9555
<null>	Amanda	Winter	amanda.winter@techlogic.net
103	Fred	Goodwin	fred808@gmail.com, fredandgillgoodwin@gmail.com, fred.goodwin@techlogic.net

~~1NF~~

65

Employee ID	First Name	Surname	Email
101	Judy	Robinson	judy.robinson@techlogic.net
102	James	Lewis	James.lewis@techlogic.net
105	Amanda	Winter	amanda.winter@techlogic.net
103	Fred	Goodwin	fred.goodwin@techlogic.net

1NF ✓

66

Multi-valued and multi-type fields

- **Multi-valued field:** more than occurrence of the same type of value in a field.

  Often occur in comma delimited list.

- **Multi-type field:** more than one occurrence of different types of values in a field.

- Columns which have multiple values or multiple types are indicative of poor design.

67

Station ID	Station Name	Power Megawatt Outputs
101	Davies Dam	402 MW (2015), 338 MW (2016)
102	Elms Nuclear Plant	476 MW (2015), 477 MW (2016)
103	Palm Beach Solar Power Station	355 MW (2015), 255 MW (2016)

68

First attempt at normalizing the data…

Station ID	Station Name	Year 2015 MW Output	Year 2016 MW Output
101	Davies Dam	402	338
102	Elms Nuclear Plant	476	477
103	Palm Beach Solar Power Station	355	255

Technically this table is in First Normal Form but is still poorly designed.

69

Second attempt at normalizing the data…

Station ID	Station Name	Year	Power Megawatt Output
101	Davies Dam	2015	402
101	Davies Dam	2016	338
102	Elms Nuclear Plant	2015	476
102	Elms Nuclear Plant	2016	477
103	Palm Beach Solar Power Station	2015	355
103	Palm Beach Solar Power Station	2016	255

Tip:
Make tables deep not wide.

70

Stations (Parent Table)

Station ID	Station Name
101	Davies Dam
102	Elms Nuclear Plant
103	Palm Beach Solar Power Station

(Primary Key)

Station Outputs (Child Table)

Station ID	Year	Power Megawatt Output
101	2015	402
101	2016	338
102	2015	476
102	2016	477
103	2015	355
103	2016	255

(Foreign Key)

71

Second Example

Park ID	Park Name	Weed 1	Weed 2	Weed 3
101	Redwood	English Ivy	Hemlock	<null>
102	Manchester	Japanese Climbing Fern	Chinese Wisteria	<null>
103	Congaree	Chinese Wisteria	Japanese Stilt Grass	Japanese Climbing Fern

72

(CPK)

Park ID	Park Name
101	Redwood
102	Manchester
103	Congaree

(PK)

Park ID	Weed ID
101	1
101	2
102	3
102	4
103	4
103	5
103	3

(FK) (FK)

Weed ID	Weed Name
1	English Ivy
2	Hemlock
3	Japanese Climbing Fern
4	Chinese Wisteria
5	Japanese Stilt Grass

(PK)

73

Summary of Steps

- Identify the entities:
  - Parks and Weed species.

- Determine the relationship between the two entities:

  Q1: Can a Park have one or many Weed species?
  *A single Park can have **many** Weeds.*

  Q2: Can a single Weed species be at one or many Parks?
  *A single Weed species can be at **many** Parks.*

-Since there is a many-to-many relationship we needed to build a linking table.

74

Second Normal Form

A table is in Second Normal Form if:

It is in First Normal Form and..

All non-key attributes are functionally dependent on the whole of every candidate key.

75

Candidate Key: {SupplierID, PartNumber, OrderDatetime}

Supplier ID	Part Number	Order Datetime	Supplier Phone	Quantity
1	9400444	2018-05-23	(455) 904-9555	400
1	9400444	2018-06-27	(455) 904-9555	200
1	6930499	2018-06-27	(455) 904-9555	50
2	8219888	2018-06-06	(912) 888-8888	900
2	111128493	2018-06-06	(912) 888-8888	150
3	9400432	2018-07-26	(646) 444-9001	300

76

~~2NF~~

Candidate Key: {SupplierID, PartNumber, OrderDatetime}

Supplier ID	Part Number	Order Datetime	Supplier Phone	Quantity
1	9400444	2018-05-23	(455) 904-9555	400
1	9400444	2018-06-27	(455) 904-9555	200
1	6930499	2018-06-27	(455) 904-9555	50
2	8219888	2018-06-06	(912) 888-8888	900
2	111128493	2018-06-06	(912) 888-8888	150
3	9400432	2018-07-26	(646) 444-9001	300

{SupplierID} → SupplierPhone

The non-key attribute SupplierPhone is functionally dependent on part of the candidate key (i.e. SupplierID).

77

Student ID	Subject ID	Grade Year	Subject Name	Student First Name	Student Last Name	Grade
11	801	2016	Science	Bill	Coleman	A
8	802	2016	Math	Janet	Yates	A
11	802	2016	Math	Bill	Coleman	B
8	806	2016	French	Janet	Yates	B
5	806	2015	French	Ali	Olsen	A
5	806	2016	French	Ali	Olsen	D

78

Candidate Key

Student ID	Subject ID	Grade Year	Subject Name	Student First Name	Student Last Name	Grade
11	801	2016	Science	Bill	Coleman	A
8	802	2016	Math	Janet	Yates	A
11	802	2016	Math	Bill	Coleman	B
8	806	2016	French	Janet	Yates	B
5	806	2015	French	Ali	Olsen	A
5	806	2016	French	Ali	Olsen	D

79

~~2NF~~

Candidate Key

Student ID	Subject ID	Grade Year	Subject Name	Student First Name	Student Last Name	Grade
11	801	2016	Science	Bill	Coleman	A
8	802	2016	Math	Janet	Yates	A
11	802	2016	Math	Bill	Coleman	B
8	806	2016	French	Janet	Yates	B
5	806	2015	French	Ali	Olsen	A
5	806	2016	French	Ali	Olsen	D

80

2NF ✓

Subject ID	Subject
801	Science
802	Math
803	French

(PK)

(CPK)

Subject ID	Student ID	Grade Year	Grade
801	11	2016	A
802	8	2016	A
802	11	2016	B
803	8	2016	B
803	5	2016	A
803	5	2015	D

(FK) (FK)

Student ID	First Name	Last Name
11	Bill	Coleman
8	Janet	Yates
5	Ali	Olsen

(PK)

81

Transitive Dependencies

A transitive dependency is an indirect relationship between data elements.

Where a data element refers to either an attribute or a candidate key.

$$A \rightarrow B \rightarrow C$$

then…

$$A \rightarrow C$$

82

Third Normal Form (3NF)

A table is in Third Normal Form if:

It is in Second Normal Form and...

No non-key attributes depend on an attribute(s) that is not a super key.

X -> Y

To achieve third Normal form:

We need to remove attributes that are transitively dependent on a candidate key and put them into a separate table(s).

83

~~3NF~~

Invoices						
Invoice ID	**Invoice Date**	**Amount**	**First Name**	**Last Name**	**Customer ID**	**City**
1	2017-01-01	5000	Anna	Adams	1001	Seattle
2	2017-01-01	5000	Anna	Adams	1001	Seattle
3	2016-03-28	3400	Spencer	Zhang	1002	Ohio
4	2018-04-17	1500	Craig	Holmes	1003	Yukon
5	2017-11-13	400	Jennifer	Prebble	1004	Cairns

InvoiceID → CustomerID → City

84

3NF ✓

(PK)

Customer ID	First Name	Last Name	City
1001	Anna	Adams	Seattle
1002	Spencer	Zhang	Ohio
1003	Craig	Holmes	Yukon
1004	Jennifer	Prebble	Cairns

(PK) (FK)

Invoice ID	Invoice Date	Amount	Customer ID
1	2017-01-01	5000	1001
2	2017-01-02	2500	1001
3	2016-03-28	3400	1002
4	2018-04-17	1500	1003
5	2017-11-13	400	1004

85

~~3NF~~

Golf Tournament Winners				
Tournament Name	**Tournament Year**	**Winner Name**	**Prize Value**	**Date of Birth**
Pacific Open	2019	Anna Adams	300000	18 July 1974
Pacific Open	2018	Yulia Jacobs	350000	18 July 1974
Millbrook Open	2017	Jack Brown	350000	22 May 1982
Oakland Classic	2018	Craig McCain	425000	20 April 1994
Oakland Classic	2019	Craig McCain	450000	20 April 1994

{TournamentName, TournamentYear} → WinnerName → DateofBirth

86

Golf Tournament Winners				
Tournament Name	**Tournament Year**	**Winner Name**	**Prize Value**	**Date of Birth**
Pacific Open	2019	Anna Adams	300000	18 July 1974
Pacific Open	2018	Yulia Jacobs	350000	18 July 1974
Millbrook Open	2017	Jack Brown	350000	22 May 1982
Oakland Classic	2018	Craig McCain	425000	20 April 1994
Oakland Classic	2019	Craig McCain	450000	20 April 1994

87

Golf Tournament Winners				
Tournament ID	**Tournament Year**	**Player ID**	**Prize Value**	**Date of Birth**
102	2019	4	300000	18 July 1974
102	2018	6	350000	18 July 1974
103	2017	2	350000	22 May 1982
101	2018	5	425000	20 April 1994
101	2019	5	450000	20 April 1994

88

(CPK)

Golf Tournaments	
Tournament ID	**Tournament Name**
101	Oakland Classic
102	Pacific Open
103	Millbrook Open

(PK)

Golf Tournament Winners			
Tournament ID	**Year**	**Player ID**	**Prize**
102	2019	4	300000
102	2018	6	350000
103	2017	2	350000
101	2018	5	425000
101	2019	5	450000

(FK) (FK)

Golf Players		
Player ID	**Name**	**Date of Birth**
1	Eric Nobel	18 July 1974
2	Jack Brown	18 July 1974
3	Eric Nobel Jnr	22 May 1982
4	Anna Adams	20 April 1994
5	Craig McCain	20 April 1994
6	Yulia Jacobs	18 July 1974

(PK)

89

Boyce-Codd Normal Form (BCNF)

Definition:
A table is in BCNF if for any non-trivial functional dependency such as X -> Y then X is a super key.

Explanation:
A table is in BCNF if each determining attribute(s) is a super key *i.e. each determining attribute (determinant) has unique values.*

A determinant is an attribute or set of attributes on the left-hand side of the functional dependency:
X -> Y

A table is in BCNF if every determinant has unique values.

90

Shipments			
Supplier ID	**Supplier Name**	**Part ID**	**Quantity**
101	Riverbridge	P1	500
101	Riverbridge	P2	300
101	Riverbridge	P3	200
103	Omegacom	P5	450
104	Vine corp	P1	100
104	Vine corp	P5	50

Candidate Keys:

{SupplierID, PartID}

{SupplierName, PartID}

SupplierID -> SupplierName

91

BCNF✓

(PK)

Supplier ID	Supplier Name
101	Riverbridge
102	XYZ ltd
103	Omegacom
104	Vine corp

(FK)

Supplier ID	Part ID	Quantity
101	P1	500
101	P2	300
101	P3	200
103	P5	450
104	P1	100
104	P5	50

(CPK)

92

Student Professors		
Student ID	**Subject**	**Professor**
101	Biology	Khloe Bellamy
101	Chemistry	Nicholas Reese
102	Biology	Andrew Jones
103	Economics	Melanie Derrick
104	Economics	Melanie Derrick
104	English	Denis McDonald

Note:
Each professor can only teach one subject.

It is possible for one subject be taught by multiple professors.

{StudentID, Subject} → Professor

Professor → Subject

93

Remembering the Normal Forms

Attributes in a table should depend on the key, the whole key, and nothing but the key, so help me Codd!

Normal Form Level	Part of Saying	Explanation
First Normal Form	*…depend on the key*	The main condition for a table to meet 1NF is that needs a key.
Second Normal Form	*the whole key*	All non-key attributes should be functionally dependent on the whole of every key.
Third Normal Form	*nothing but the key*	All non-key attributes should depend on nothing but the key.
Boyce-Codd Normal Form	*nothing but the key*	All attributes should depend on nothing but the key.

94

3NF vs BCNF

X -> Y

In third normal form if the right-hand side attribute(s) is prime then we don't care if the left-hand side is a super key.

However, in Boyce-Codd normal form, the left-hand side of a functional dependency always has to be a super key regardless of whether the right-hand side is a prime or non-prime attribute(s).

95

Multi-valued dependencies

A multi-valued dependency occurs when the following conditions are met:

There are at least 3 attributes (e.g. X, Y, Z) where 2 of the attributes are independent of each other and both are dependent on a third attribute.

For a dependency X →→ Y, for every value of X there is a set of well-defined values for Y.

X→→ Z, for every value of X there is a set of well-defined values for Z.

Y is independent of Z

96

Fourth Normal Form (4NF)

A table is in Fourth Normal Form if…

It is in Boyce-Codd Normal Form and…

For every non-trivial multi-valued functional dependency X ->> Y then X is a super key.

97

Multi-valued Dependency Example

University Courses		
Course	**Course Book**	**Lecturer**
ECON101	Econ Principles	A. Miller
ECON101	Microeconomics	A. Miller
ECON101	Econ Principles	F. Jacobs
ECON101	Microeconomics	F. Jacobs
ECON200	Econ Principles	A. Miller
ECON200	Econ Principles	L. Brown

~~**4NF**~~

Course ->-> Book
Course ->-> Lecturer

Above is interpreted as…

Course determines multiple values for **CourseBook**
and
Course determines multiple values for **Lecturer**
and
CourseBook *is independent of* ***Lecturer.***

98

University Courses		
Course	**Book**	**Lecturer**
ECON101	Econ Principles	A. Miller
ECON101	Microeconomics	A. Miller
ECON101	Econ Principles	F. Jacobs
ECON101	Microeconomics	F. Jacobs
ECON200	Econ Principles	A. Miller
ECON200	Econ Principles	L. Brown

Course ->-> Book
Course ->-> Lecturer

We want to add a new textbook (Freakonomics) for the ECON101 course.
How many rows do we need to add to the table?

University Courses		
Course	**Book**	**Lecturer**
ECON101	Econ Principles	A. Miller
Microeconomics	Microeconomics	A. Miller
ECON101	**Freakonomics**	**A. Miller**
ECON101	Econ Principles	F. Jacobs
ECON101	Microeconomics	F. Jacobs
ECON101	**Freakonomics**	**F. Jacobs**
ECON200	Econ Principles	A. Miller
ECON200	Econ Principles	L. Brown

We would need to add 2 new records to the table: one for each of the lecturers teaching the ECON101 course.

99

(CPK)

Course Books	
Course	**Book**
ECON101	Econ Principles
ECON101	Microeconomics
ECON101	Freakonomics
ECON200	Econ Principles

(FK)

(CPK)

Course Lecturers	
Course	**Lecturer**
ECON101	A. Miller
ECON101	F. Jacobs
ECON200	A. Miller
ECON200	L. Brown

(FK)

Courses	
Course	**Course Name**
ECON101	Introduction to Economics
ECON200	Level 2 Economics

(PK)

Other tables would include:

Books – table of unique books names.

Lecturers – table of unique lecturer names.

100

Types of data anomalies

What can go wrong when data structures are not normalized

101

Update Anomalies

Instructors		
Instr ID	**First Name**	**Last Name**
1	Chris	Jackson
2	Adam	Richardson
3	Gillian	Singh

Classes				
Instr ID	**Class ID**	**Class Name**	**Contact**	**Cost**
1	101	Intro to SQL	808-1980	5000
2	101	Intro to SQL	808-1980	5000
1	102	Intro to Java	808-7812	5500
3	103	Intro to PowerPivot	808-7809	4500

102

Update Anomalies

Instructors		
Instr ID	**First Name**	**Last Name**
1	Chris	Jackson
2	Adam	Richardson
3	Gillian	Singh

Classes				
Instr ID	**Class ID**	**Class Name**	**Contact**	**Cost**
1	101	Intro to SQL	808-1980	5000
2	101	Intro to SQL	808-1980	5000
1	102	Intro to Java	808-7812	5500
3	103	Intro to PowerPivot	808-7809	4500
3	101	Intro to SQL	808-1981	5000

103

Insert Anomalies

Instructors		
Instr ID	**First Name**	**Last Name**
1	Chris	Jackson
2	Adam	Richardson
3	Gillian	Singh

Classes				
Instr ID	**Class ID**	**Class Name**	**Contact**	**Cost**
1	101	Intro to SQL	808-1980	5000
2	101	Intro to SQL	808-1980	5000
1	102	Intro to Java	808-7812	5500
3	103	Intro to PowerPivot	808-7809	4500

(CPK)

Instr ID	Class ID	Class Name	Contact	Cost
<null>	104	Advanced Excel	808-8543	3999

104

Deletion Anomalies

Instructors		
Instr ID	**First Name**	**Last Name**
1	Chris	Jackson
2	Adam	Richardson
~~3~~	~~Gillian~~	~~Singh~~

Classes				
Instr ID	**Class ID**	**Class Name**	**Contact**	**Cost**
1	101	Intro to SQL	808-1980	5000
2	101	Intro to SQL	808-1980	5000
1	102	Intro to Java	808-7812	5500
~~3~~	~~103~~	~~Intro to~~ PowerPivot	~~808-7809~~	~~4500~~

(CPK)

105

Solution

Instructors		
Instr ID	**First Name**	**Last Name**
1	Chris	Jackson
2	Adam	Richardson
3	Gillian	Singh

(PK)

Instructor Classes	
Instr ID	**Class ID**
1	101
2	101
1	102
3	103

(FK) *(FK)*

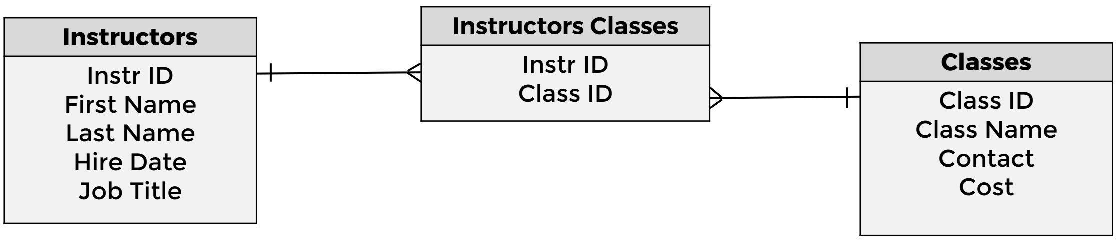
Composite Primary key (PK)

Classes			
Class ID	**Class Name**	**Contact**	**Cost**
101	Intro to SQL	808-1980	5000
102	Intro to Java	808-7812	5500
103	Intro to PowerPivot	808-7809	4500

(PK)

106

Solution as an Entity Relationship Diagram (ERD)



107

Naming Conventions

- No right or wrong naming of columns, tables, etc.
- Personal Recommendations:
  - Use plurals for table names i.e. PRODUCTS instead of PRODUCT.
  - Give tables and columns meaningful names.
  - Avoid adding spaces in table and column names. Instead use underscores.
  - If possible, give a foreign key column the same name as the primary key column that it references.
  - Focus on consistent naming conventions.

108

Data Types

- Many data types are supported by SQL Server.
- The main classes of data types include:
  - **Character:**
    - Fixed Length
    - Varying Length
  - **Number:**
    - Integer (whole number) types
    - number types to a specific number of decimal places.
    - Floating point
  - **Date:**
    - With time component
    - With time zone offset

109

PROJECT_INFO							
Project ID	**Project Name**	**Employee ID**	**Employee Name**	**Department ID**	**Department Name**	**Salary**	**Project Budget**
101	Digital Services	1	B. Jones	1005	IT	80000	25000
101	Digital Services	2	C. Snow	1004	Marketing	90000	25000
102	HR system	1	B. Jones	1005	IT	80000	30000
102	HR system	3	W. Glass	1004	Marketing	100000	30000
102	HR system	4	S. Singh	1006	HR	90000	30000

110

Decomposition Exercise Steps

- Open and execute "Decomposition Exercise.sql" file in SSMS to create the PROJECT_INFO table.
- Identify the entities in the PROJECT_INFO table
- Determine the relationship type between each pair of entities.
- Write the create table statements for the new tables. Every table should have a primary key.
- Populate the new tables with data extracted from the original PROJECT_INFO table.

111

Decomposition Exercise Solution

- Identify the entities in the PROJECT_INFO table:
  - Projects
  - Employees
  - Departments

112

Determine the Relationship Type between Entities

Projects and Employees:

- Can one **project** be associated with one or many **employees**?
  One project can have ***many*** *employees.*
- Can one **employee** be associated with one or many **projects**?
  One employee can work on ***many*** *projects.*

Therefore we have a many-to-many relationship between employees and projects.

EmployeesProjects
EMPLOYEE_ID
PROJECT_ID

113

Determine the Relationship Type between Entities

Projects and Departments:

There is no direct relationship between project and departments.

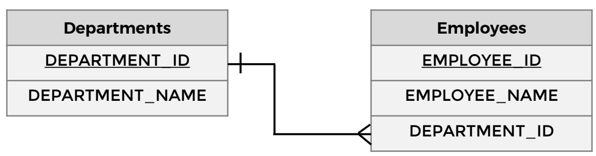
114

Determine the Relationship Type between Entities

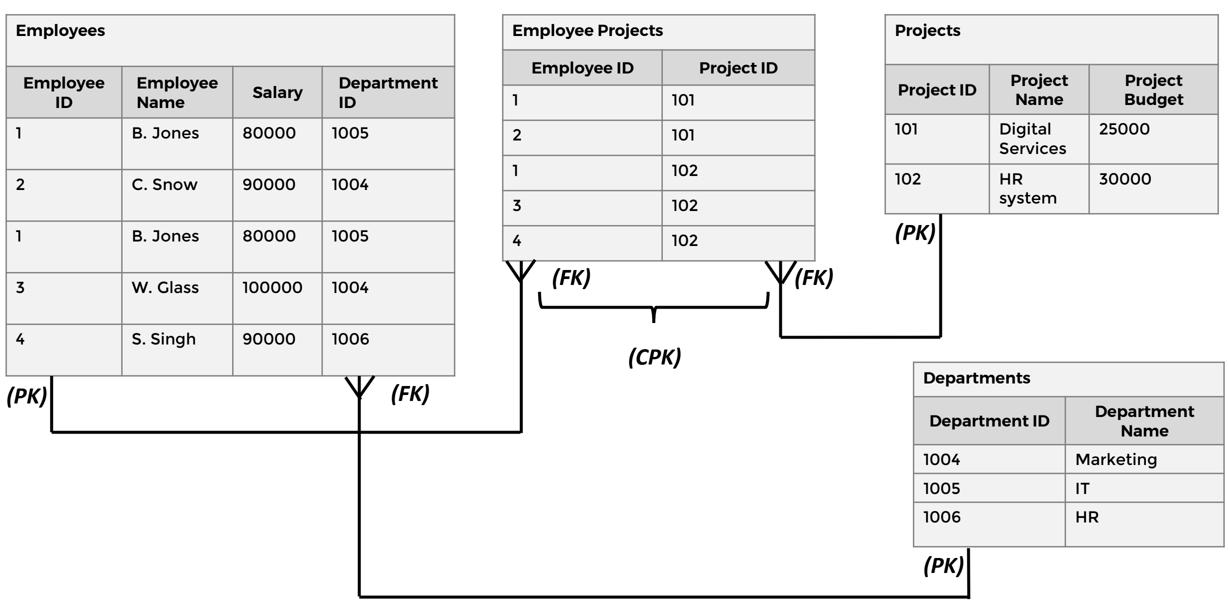
Departments and Employees:

- Can one **department** be associated with one or many **employees**?
  *One department can have **many** employees.*
- Can one **employee** be associated with one or many **departments**?
  *One employee can only be associated with **one** department.*

Therefore we have a one-to-many relationship between departments and employees:



115



Employees

Employee ID	Employee Name	Salary	Department ID
1	B. Jones	80000	1005
2	C. Snow	90000	1004
1	B. Jones	80000	1005
3	W. Glass	100000	1004
4	S. Singh	90000	1006

Employee Projects

Employee ID	Project ID
1	101
2	101
1	102
3	102
4	102

Projects

Project ID	Project Name	Project Budget
101	Digital Services	25000
102	HR system	30000

Departments

Department ID	Department Name
1004	Marketing
1005	IT
1006	HR

116

Library Customers		
Person ID	**First Name**	**Last Name**
101	Judy	Jackson
102	William	Davis
103	Sarah	Romano
104	Jerry	Kumar
105	Craig	Todd

(PK)

Library Books	
ISBN	**Book Title**
978-0670033041	East of Eden
978-1250303882	Five Ingredients
978-0062390622	The Alchemist
978-1787016965	The Travel Atlas
978-0141392462	The Count of Monte Cristo
978-0618640157	The Lord of the Rings
978-0446568814	Rich Dad Poor Dad
978-0060752613	The Intelligent Investor

(PK)

Book Checkouts Report				
Person ID	**Book Title 1**	**Book 1 Due Date**	**Book Title 2**	**Book2 Due Date**
101	East of Eden	20/7/2020	Five Ingredients	20/7/2020
101	The Alchemist	20/7/2020	The Travel Atlas	20/7/2020
102	The Count of Monte Cristo	15/8/2020	The Lord of the Rings	17/8/2020
103	Rich Dad Poor Dad	19/7/2020		
104	The Intelligent Investor	13/8/2020		

- Open and execute the “Library Database Exercise.sql” file in SSMS to create the tables shown here.
- Create a new table called BOOK_CHECKOUTS. This table is to be a normalized table that replaces the BOOK_CHECKOUTS_REPORT table. Ensure that the new table has an appropriate primary key and foreign keys.

Advanced exercise:
Write a query to extract data from the old BOOK_CHECKOUTS_REPORT table as well as the ISBN from the LIBRARY_BOOKS table and populate the new BOOK_CHECKOUTS table with this data.

117

Library Customers		
Person ID	**First Name**	**Last Name**
101	Judy	Jackson
102	William	Davis
103	Sarah	Romano
104	Jerry	Kumar
105	Craig	Todd

(PK)

Library Books	
ISBN	**Book Title**
978-0670033041	East of Eden
978-1250303882	Five Ingredients
978-0062390622	The Alchemist
978-1787016965	The Travel Atlas
978-0141392462	The Count of Monte Cristo
978-0618640157	The Lord of the Rings
978-0446568814	Rich Dad Poor Dad
978-0060752613	The Intelligent Investor

(PK)

Book Checkouts Report				
Person ID	**Book Title 1**	**Book 1 Due Date**	**Book Title 2**	**Book2 Due Date**
101	East of Eden	20/7/2020	Five Ingredients	20/7/2020
101	The Alchemist	20/7/2020	The Travel Atlas	20/7/2020
102	The Count of Monte Cristo	15/8/2020	The Lord of the Rings	17/8/2020
103	Rich Dad Poor Dad	19/7/2020		
104	The Intelligent Investor	13/8/2020		

118

Library Customers		
Person ID	**First Name**	**Last Name**
101	Judy	Jackson
102	William	Davis
103	Sarah	Romano
104	Jerry	Kumar
105	Craig	Todd

(PK)

Library Books	
ISBN	**Book Title**
978-0670033041	East of Eden
978-1250303882	Five Ingredients
978-0062390622	The Alchemist
978-1787016965	The Travel Atlas
978-0141392462	The Count of Monte Cristo
978-0618640157	The Lord of the Rings
978-0446568814	Rich Dad Poor Dad
978-0060752613	The Intelligent Investor

(PK)

Book Checkouts		
Person ID	**ISBN**	**Due Date**
101	978-0670033041	20/7/2020
101	978-0062390622	20/7/2020
102	978-0141392462	15/8/2020
103	978-0446568814	19/7/2020
104	978-0060752613	13/8/2020
101	978-1250303882	20/7/2020
101	978-1787016965	20/7/2020
102	978-0618640157	17/8/2020

(FK) ***(FK)***

Composite Primary key (CPK)

119