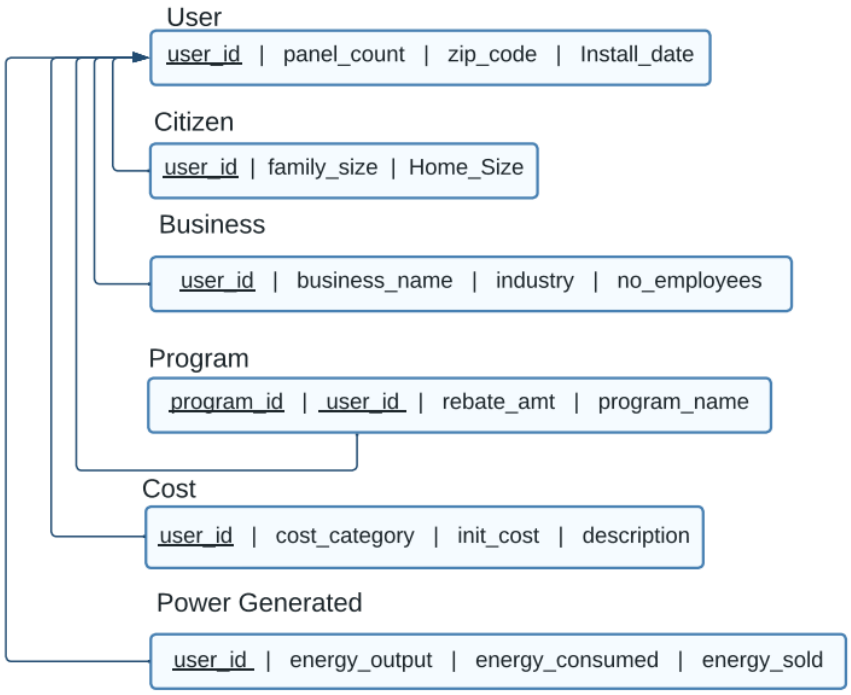
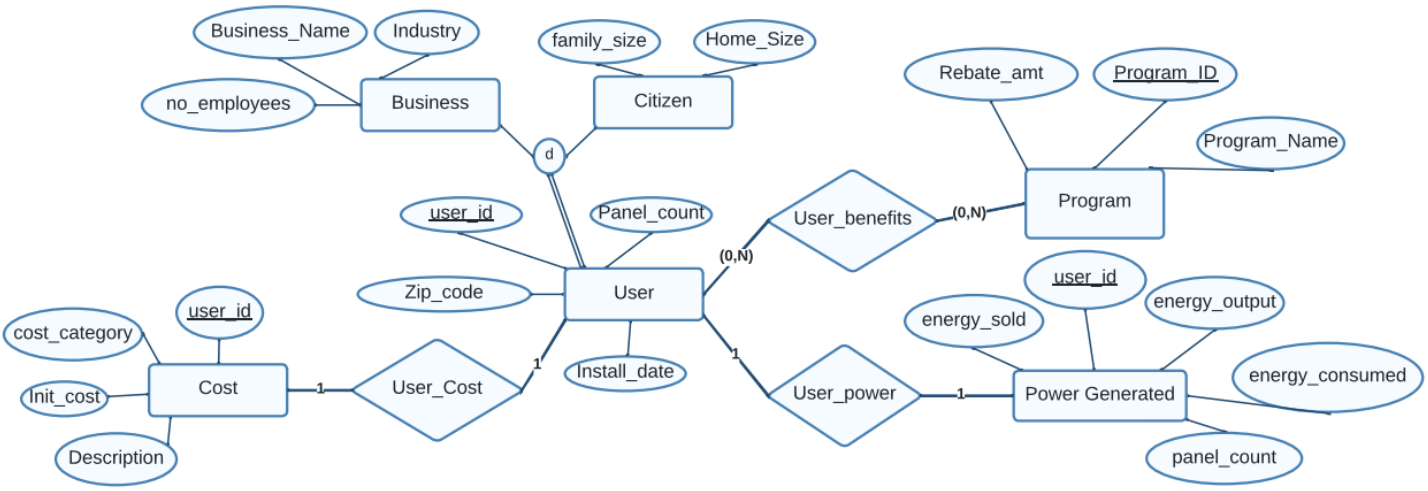
Phase IV: Elaborated Database Design

By: Claire Engebreth, Simon Blamo, Joe Petrecca



Updated Database Schema



Updated Database ER Diagram

Boyce-Codd Normal Form

First Normal Form: Our database is in 1NF because no multi-valued attributes exist.

Second Normal Form: Our database is in 2NF because 1) it is in 1NF and 2) there are no non-key attributes dependent on full primary keys.

Third Normal Form: There are no transitive functional dependencies in our database and our database is in 2NF, therefore our database is also in 3NF.

Boyce-Codd Normal Form: Our database is in BCNF because every attribute in a relation functionally depends on the primary key, not other non-key attributes.

USER Relation: The USER relation has a primary key (USER_ID), and all other attributes are fully functionally dependent on the primary key. Therefore, it satisfies the requirements for BCNF.

CITIZEN relation: The CITIZEN relation has a foreign key (USER_ID) that references the primary key in the USER relation, and all other attributes are fully functionally dependent on the foreign key. Therefore, it satisfies the requirements for BCNF.

BUSINESS relation: The BUSINESS relation has a foreign key (USER_ID) that references the primary key in the USER relation, and all other attributes are fully functionally dependent on the foreign key. Therefore, it satisfies the requirements for BCNF.

COST relation: The COST relation has a foreign key (USER_ID) that references the primary key in the USER relation, and all other attributes are fully functionally dependent on the foreign key. Therefore, it satisfies the requirements for BCNF.

POWER GENERATED relation: The POWER GENERATED relation has a foreign key (USER_ID) that references the primary key in the USER relation, and all other attributes are fully functionally dependent on the foreign key. Therefore, it satisfies the requirements for BCNF.

PROGRAM relation: The PROGRAM relation has a primary key (PROGRAM_ID), but the REBATE_AMT attribute is functionally dependent on the PROGRAM_ID primary key because every program has a distinctive rebate amount rather than each user having a distinctive rebate amount.

VIEWS

Customer View:

This view can be used to retrieve data for customers, including their user ID, panel count, zip code, and installation date if they are a business then their business information, and if they are a citizen then their customer data.

Data requirements: USER_ID, PANEL_COUNT, ZIP_CODE, INSTALL_DATE

Transaction requirements: SELECT statement

Example Query:

A query that retrieves the user ID, panel count, and zip code for customers who installed solar panels between January 1, 2022, and December 31, 2022. It does so by selecting data from the virtual table called CUSTOMER_VIEW, which has been created based on a query that retrieves data from the USER table in the database. The query includes a WHERE clause that filters the results to only include customers who installed panels during the specified time frame.

SQL Query:

```
CREATE VIEW CUSTOMER_BASIC_VIEW AS

SELECT USER_ID, PANEL_COUNT, ZIP_CODE, INSTALL_DATE

FROM USER

WHERE USER_ID = '12345';
```

Energy View:

This view can be used to retrieve data for energy generation and consumption, energy output, energy consumed, and energy sold for different users.

Data requirements: USER_ID, PANEL_COUNT, ZIP_CODE, INSTALL_DATE

Transaction requirements: SELECT statement

Example Query:

A query that retrieves the total energy output and energy consumption for each user in the database. It groups the data by user ID and uses the aggregate function SUM to calculate the totals of energy outputted and consumed.

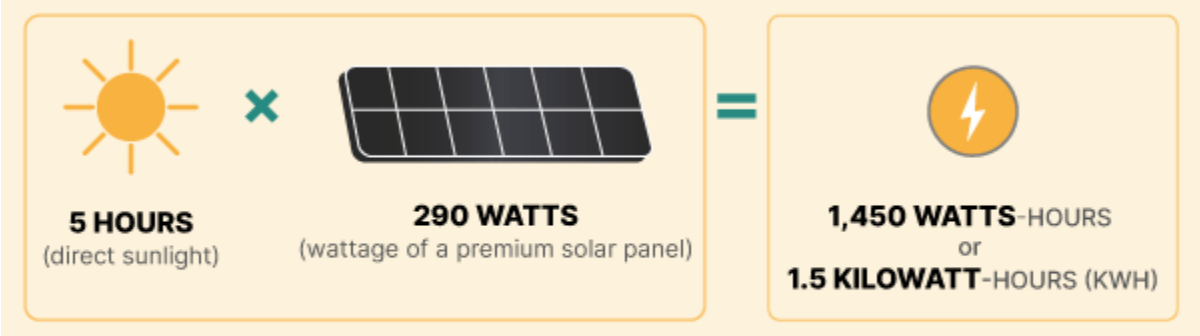
SQL Query:

```
CREATE VIEW ENERGY_VEW AS

SELECT (USER_ID) SUM(ENERGY_OUTPUT) AS
TOTAL_ENERGY_OUTPUT

SUM(ENERGY_CONSUMED) AS TOTAL_ENERGY_CONSUMED

FROM ENERGY_VIEW GROUP BY USER_ID;
```



Cost by Zip Code View:

This view can be used to retrieve data for solar panel costs by zip code, including the zip code and total cost.

Data requirements: ZIP_CODE, TOTAL_COST.

Transaction requirements: SELECT statement.

Example Query:

A query that retrieves the total solar panel cost for all users with a specific zip code. It does so by selecting the initial cost data from the COST table in the database using the aggregate SUM function. The WHERE clause filters the results to only include data for users with the specified zip code, which is obtained from the USER table in the database. This query uses a subquery to obtain the user IDs for all users with the specified zip code and then uses those user IDs to filter the initial cost data in the COST table. To use this query in an application, you can prompt the user to enter their zip code, and then substitute the user's input into the query using string formatting or a parameterized query.

SQL Query:

```
CREATE VIEW COST_BY_ZIPCODE_VIEW AS

SELECT SUM(INIT_COST) AS TOTAL_COST

FROM COST

WHERE USER_ID IN (

   SELECT USER_ID

   FROM USER

   WHERE ZIP_CODE = '08618'

);

GROUP BY ZIP_CODE;
```