Note: See Git repo for all artifacts, code, and demo walkthrough video

Git repo: https://github.com/brockgion/ixport (Note: all SQL Queries in ixport/queries)

Abstract

The process for homeowners to connect residential solar systems to the electrical grid is difficult to understand, creating a significant barrier to renewable energy adoption. This project directly addresses this problem by designing and implementing a database-driven web portal to model and track the interconnection application (IX) lifecycle. The core component of the project is a relational database, designed using Entity-Relationship (ER) modeling and implemented in PostgreSQL, which serves as a single source of truth for all application data. A user-facing web application allows homeowners and installers to submit applications and view their real-time status as they progress through a simplified, five-step workflow. This work successfully demonstrates how foundational database principles—including normalized schema design, SQL DDL for data integrity, and query-driven application logic—can be applied to solve a real-world information problem. This is a proof of concept - its primary objective is to show how interconnection applications can be tracked through a user-friendly experience, thereby empowering homeowners with rooftop solar installations.

Introduction

The integration of distributed renewable energy sources, particularly residential solar systems, remains one of the most significant challenges to achieving large-scale decarbonization of the electrical grid. As Gorman et al. [1] describe, grid connection barriers have slowed the adoption of clean energy technologies, not because of generation limitations but due to administrative and procedural bottlenecks. Similarly, Schweitzer et al. [2] highlight how interconnection queues and hosting-capacity constraints have created inefficiencies in bringing distributed energy resources (DERs) online. Many utility-managed interconnection systems remain proprietary and opaque, as noted by Valova and Brown [3], preventing customers and installers from tracking project progress. This project models a transparent, database-driven interconnection portal that allows customers, installers, and utility administrators to view the application lifecycle in real time. The system provides a five-step workflow—site selection, submission, approval, construction, and completion—reflecting both academic best practices and practical database design principles. Drawing from Chelminski [4] and Knuth and Ventrella [5], the project emphasizes how data transparency and governance are central to creating equitable, efficient interconnection regimes that can scale with future renewable energy adoption.

- **Objective and Impact**
    - **Build a database-driven interconnection portal that models how electric residential utility customers apply to connect renewable energy systems (e.g., rooftop solar) to the electrical grid.** Unless you work for a utility company, this

process is often hard to understand so this will help educate residents (i.e homeowners) into how the process works involving utilities, installers, and customers.

- **Problem**
    - **Most interconnection portals are closed source, managed by utility companies, rigid, and do not expose how application data flows through approval stages.** This project models a simplified Interconnection Application (IX) pipeline using sound relational database principles and allows customers to see their own interconnection (IX) application status.
- **Concept**
    - **The solution models key entities: customer applicant, installer, application, solar system specs, and the utility company reviews applicants' system design.** The system enables querying and visualization of interconnection timelines and bottlenecks, making it more useful than checking static spreadsheets that are often outdated, or customers sending emails to utilities and not getting a timely response.

Course Relevance

Course Topic 1: ER Modeling and Relational Mapping (Topics in Weeks 1-5)

This project is built on core ER Modeling and Relational Mapping concepts. The Entity-Relationship diagram reflects real-world interconnection stakeholders—customers, premises, utility services, installers, and applications (Appendix Figures A1, A2). The final design implements nine relationships with varying cardinality (Appendix Figure A6). For example, there's one-to-many (a customer submitting multiple applications), optional (customers may lack assigned premises), and strict one-to-one (systems to applications via UNIQUE constraint on the foreign key). Participation constraints enforce business rules—each application requires exactly one customer and installer.

Topic 2: SQL DDL and Schema Design (Weeks 6,7)

The implementation relies extensively on SQL Data Definition Language (DDL) from Chapter 7. Each entity uses CREATE TABLE statements with defined primary and foreign keys. The interconnection_application table includes a UNIQUE constraint on ix_system_id enforcing the one-to-one relationship between systems and applications, matching both domain logic and course cardinality principles. UUID keys provide globally unique identifiers. NOT NULL constraints enforce mandatory participation. The schema implements indexing on frequently searched fields (customer names, utility providers, application status) for performance optimization. This demonstrates practical application of table creation, relational integrity, and schema design principles (Appendix Figure A5).

Topic 3: Applications and Query Workflows (Week 8, Week 13 information retrieval) The

PostgreSQL-backed system demonstrates practical application of query-driven workflows and information retrieval concepts from Weeks 8 and 13. The implementation showcases several query patterns: (1) simple aggregation queries for status counts (Appendix Figure A7), (2) more complex joins using JOIN LATERAL to retrieve application timelines with completion dates across all stages (Appendix Figure A8), and (3) filtered retrieval queries that allow installers to view only their assigned applications (Appendix Figure A9). The application_status_history table enables temporal queries that track state transitions over time, supporting audit trails and bottleneck analysis—a practical application of information retrieval principles where historical data informs operational decisions. Indexed columns on customer names, utility providers, and application status optimize query performance (Appendix Figure A5).

Implementation

Diagram Architecture Overview

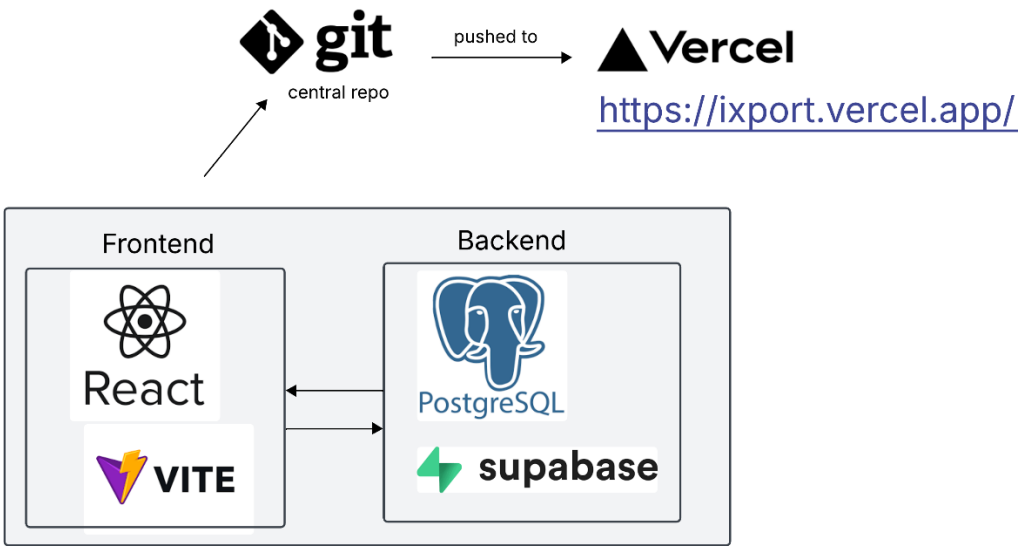


Figure 1: Diagram overview of key components of the web application. PostgreSQL is the underlying database technology.

Interconnection Portal Overview

Key components of this project are PostgreSQL + React. **See Git Repo file: App.jsx**. This main file is the single-file React app used for a Solar Interconnection Portal that tracks customer applications and approval stages.

See web app for interactive demo, showing full implementation:

- https://ixport.vercel.app/

Frontend

- React + Vite — lightweight, fast development environment for the single-page web app.
- Tailwind CSS — utility-first styling for responsive, minimal UI with Lucide React SVG icons
- JavaScript — handles logic for forms, workflows, and UI state (e.g. expand/collapse)

Backend / Database

- PostgreSQL (using Supabase) — Fully managed Postgres database hosting with authentication and real-time updates.
- PostgreSQL Schema Design — normalized structure
- UUID Primary Keys — for globally unique records across entities.
- Triggers & Functions — automate timestamps and log application progress history

Architecture & Logic

- Frontend-driven workflow logic — defines all status steps (site_selection → submitted → approved → construction → complete).
- Dynamic timeline UI — status icons, step numbers, and timestamps updated live.
- Notes & history tracking — editable notes per stage, saved back to the database.
- Expand/Collapse Cards — smooth UI state management with React hooks.
- Admin vs Customer view — toggles between full management and read-only modes.

Integration Flow

- User creates a new application → inserts related records across multiple tables (account, customer, premise, system, installer, application).
- A PostgreSQL trigger automatically logs status changes to application_status_history. After each change, the frontend re-fetches data to display updated state.

CI/CD

- Integration between Github paired with Vercel allows instant updates
- Changes automatically deployed on git push

Results

The interconnection portal successfully demonstrates a functional database-driven system for managing solar applications.

The implementation provides two distinct interfaces: a customer-facing view showing application status in real-time and an administrative interface enabling workflow management with stage advancement and note-taking capabilities (Figure 2).

The five-step workflow (site selection → submitted → approved → construction → complete) simplifies the traditionally complex interconnection process (Appendix Figure A3). The application_status_history table logs all state transitions, creating an auditable trail.

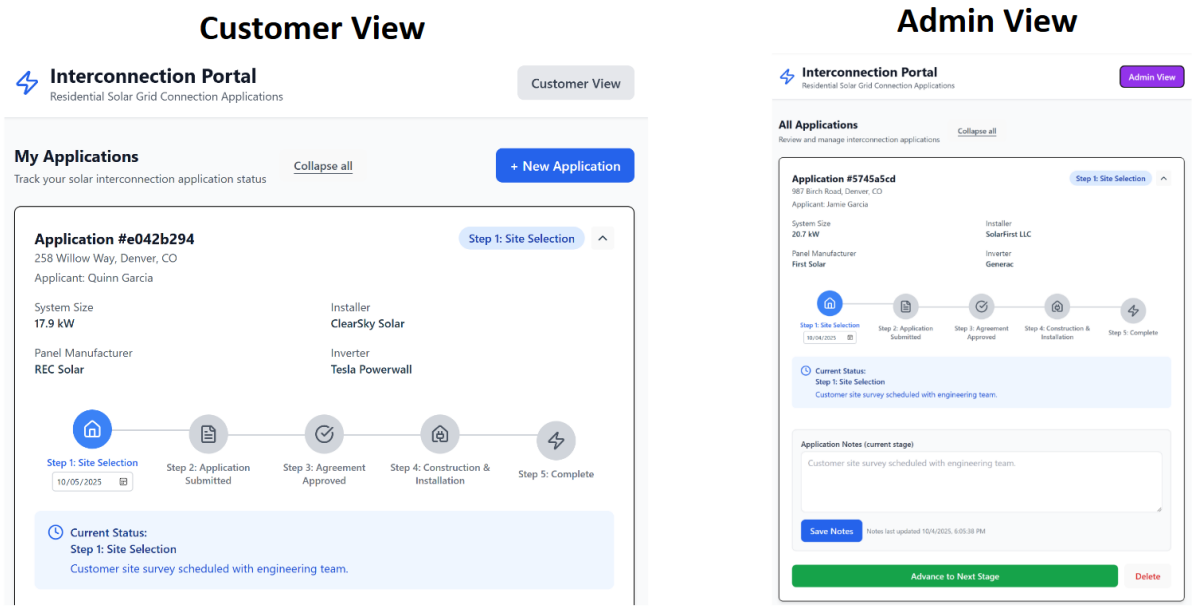


Figure 2: Customer view vs. Admin view of an Interconnection App (Note: all dummy info, not real)

The system eliminates the need for customers to email utilities or check outdated spreadsheets. All application data is accessible and viewable online at a glance. As an admin, it's easy to see all applications in the portal (Figure 3).

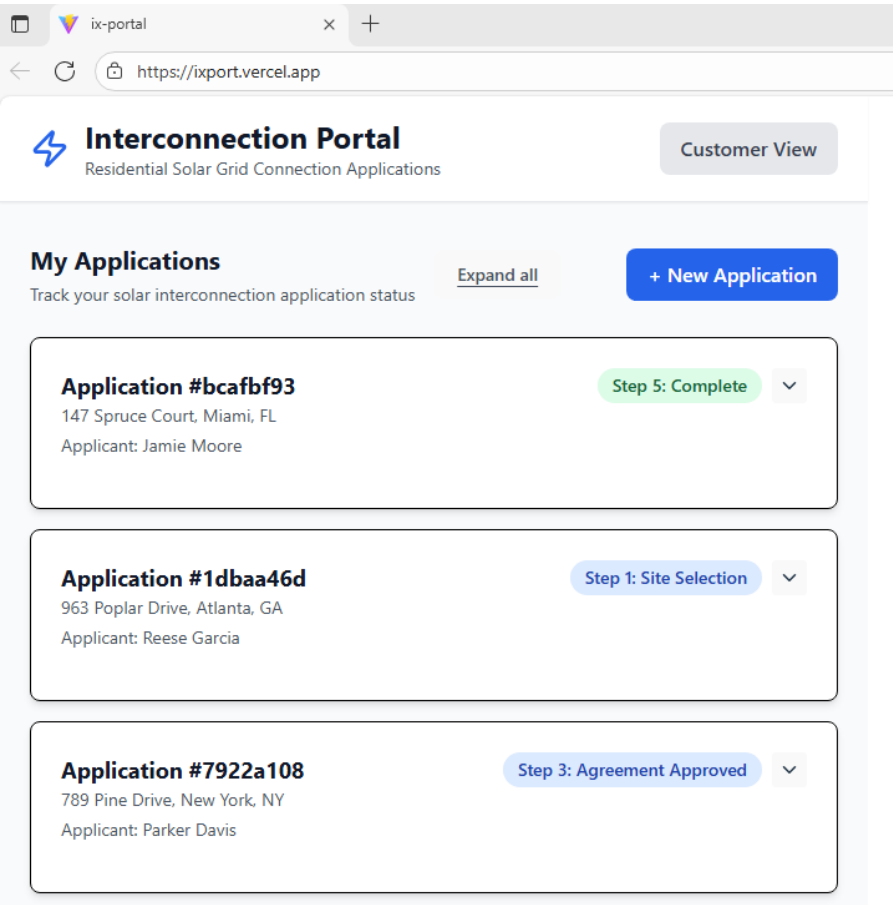


Figure 3: Example of how applications appear as they enter into the queue with associated status of each one. (Green = Complete, Blue = In Progress)

Overall, the system enables efficient information access through both a web interface and indexed backend queries (See Appendix A7, A8, A9 –examples of 3 different queries).

Ethics

This project raises important ethical considerations. These are three examples:

1) **Customer privacy** is critical—applications contain sensitive PII including addresses and electrical system details. The database requires role-based access control ensuring customers view only their own applications while administrators access all records. Protecting that data requires strict role-based access control, ensuring that customers only see their own application records while administrators have broader but audited privileges.
2) **Fair Access and Business Responsibility** — Ethical business practices require fairness in how data is shared and viewed. Solar installers, for example, should have access only to the projects they manage—not to competitor data that could reveal pricing, timelines, or customer details. As Valova and Brown [3] note, distributed energy systems rely on collaboration between multiple stakeholders.
3) **Infrastructure planning ethics** addresses data accuracy. Utilities and planners rely on interconnection data for grid capacity forecasting and infrastructure investments as discussed by Gorman et al. [1] and Chelminski [4], poor or incomplete interconnection data can delay renewable energy expansion and lead to inequitable outcomes in grid access. The system's integrity constraints (NOT NULL, foreign keys) ensure baseline accuracy, but human entry errors would remain *very* concerning. Production systems need validation rules and quality checks to maintain trustworthiness for infrastructure planning decisions.

Conclusion

This project shows how ER modeling, normalized schemas, and SQL DDL solve real problems in renewable energy. I applied core course concepts—ER-to-relational mapping, data integrity constraints, and query-driven workflows—to build a working system that tracks solar interconnection applications from start to finish.

It's important to note this is a proof-of-concept. For real-world deployment it’s nowhere near complete. It would need better security, proper authentication, role management, integration with actual utility systems, and regulatory compliance features. Future improvements could include more detailed system specs, automated email notifications when status changes, and analytics to identify where applications get stuck in the queue.

This project shows how a database-driven portal can make the interconnection process more transparent and manageable for homeowners. Instead of sending emails to utilities or checking outdated spreadsheets, customers can see their application status in real-time! As more people adopt solar energy, systems like this become essential for managing the growing number of grid connections.

References

References including at least 5 peer-reviewed publications that were cited in the text

1. Gorman, W., Mills, A. D., Seel, J., & O'Shaughnessy, E. (2024). Grid connection barriers to renewable energy deployment in the United States. Joule, 9(2), 101791. https://www.cell.com/action/showPdf?pii=S2542-4351%2824%2900503-8

2. Schweitzer, E., McDermott, T. E., & Radatz, P. (2024). Evaluating interconnection queue impacts using hosting capacity analysis. In 2024 IEEE Power & Energy Society General Meeting (PESGM) (pp. 1–5). IEEE. https://doi.org/10.1109/PESGM51994.2024.10688458

3. Valova, R., & Brown, G. (2022). Distributed energy resource interconnection: An overview of challenges and opportunities in the United States. Solar Compass, 2, 100021. https://doi.org/10.1016/j.solcom.2022.100021
4. Chelminski, K. (2025). Sparking adaptation: The politics of reforming effective interconnection regimes in Massachusetts and New York. Energy Research & Social Science, 127, 104125. https://doi.org/10.1016/j.erss.2025.104125
5. Knuth, S., & Ventrella, J. (2025). Renewables in the queue: capital landing and the present crisis in power transmission. Finance and Space, 2(1), 77–94. https://doi.org/10.1080/2833115X.2025.2481071

Appendix

Figure A1: First Iteration Relational Schema - 7 Key Entities

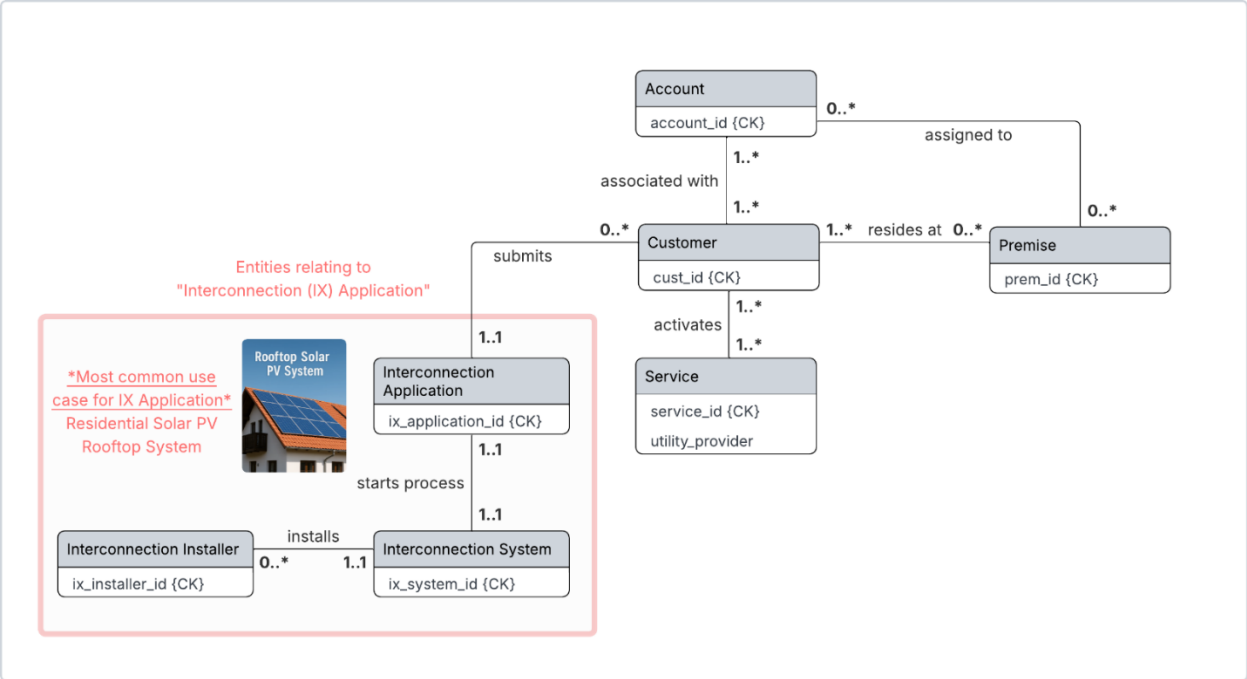


Figure A1: First iteration of diagram showing distinction between key entities. Red outline box is highlighting the “interconnected application” related entities.

7 key entities:

- Account
- Customer
- Premise
- Service
- Interconnection Application
- Interconnection System
- Interconnection Installer

Figure A2: Second Iteration of Relational Schema - 8 Key Entities

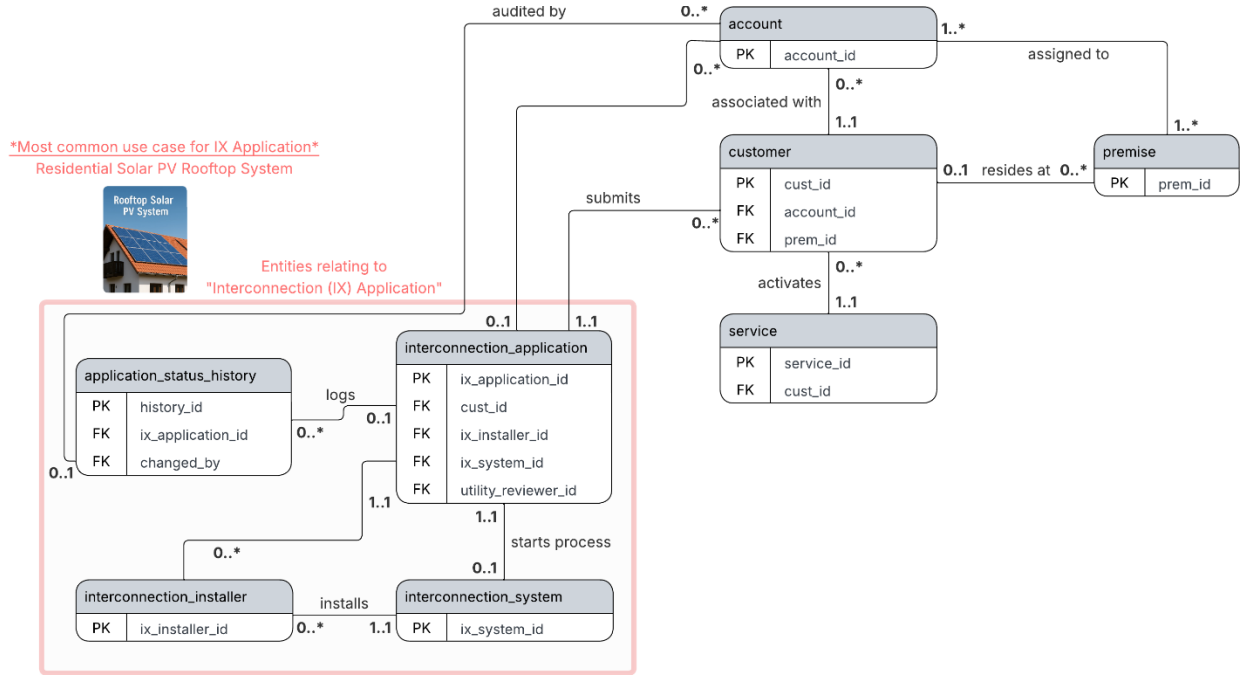
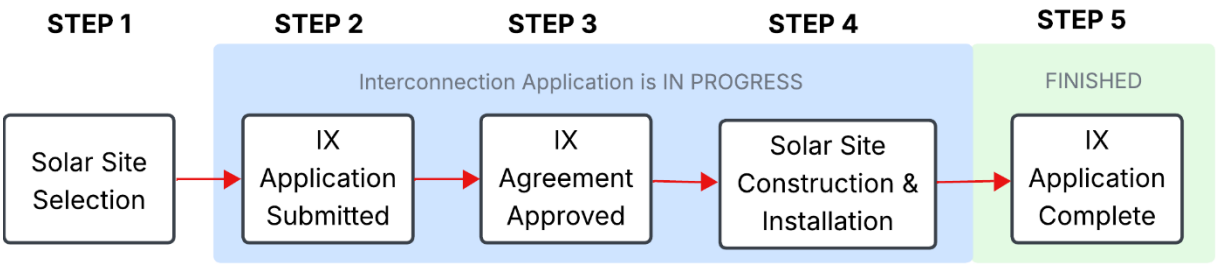


Figure A2: Second iteration diagram. Added “application_status_history” table. Further defined participation/cardinality relationships. Note: naturally this is subject to change as business requirements evolve.

Figure A3: IX App Process - Application Stages broken into 5 Main Steps



FIRST STEP: Determine the location of physical "Premise"
Customer must provide actual address of where solar panels will be installed.

LAST STEP: The Premise becomes "interconnected"
Solar panels are installed and IX application is deemed complete. Interconnection date is recorded as YYYY/MM/DD.

Figure A3: Generalized Interconnection Application (IX App) Process – 5 main steps.

Note: At any point in the process an IX application can be “withdrawn” and deemed incomplete.

Figure A4: Example Website View of Interconnection Application

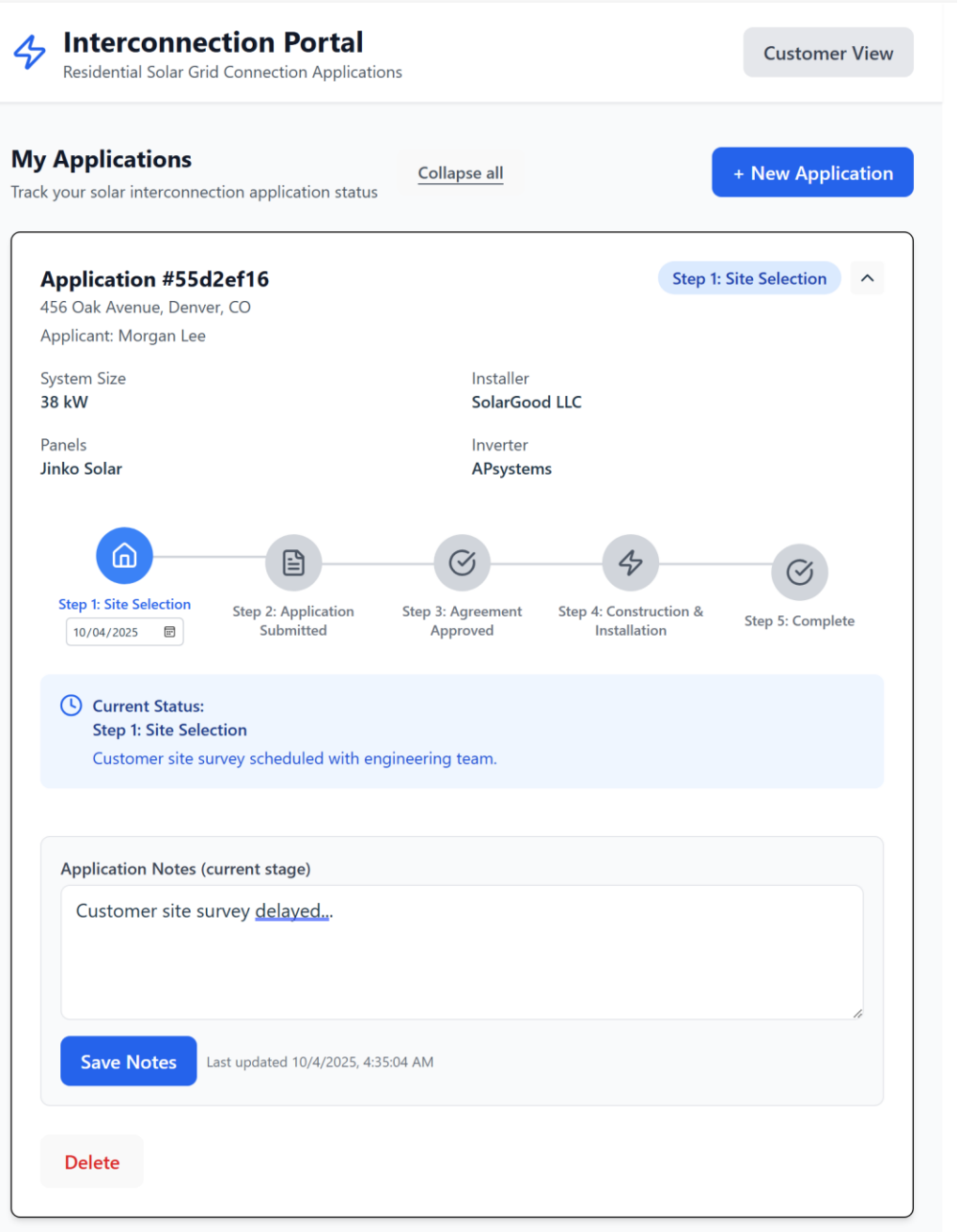


Figure A4: Example of how a customer-facing view through a web portal can show the different stages of an interconnection application. Ability for customers, solar installers, and utility providers to track real-time status updates.

Figure A5: Final relational schema – allows tracking application status history

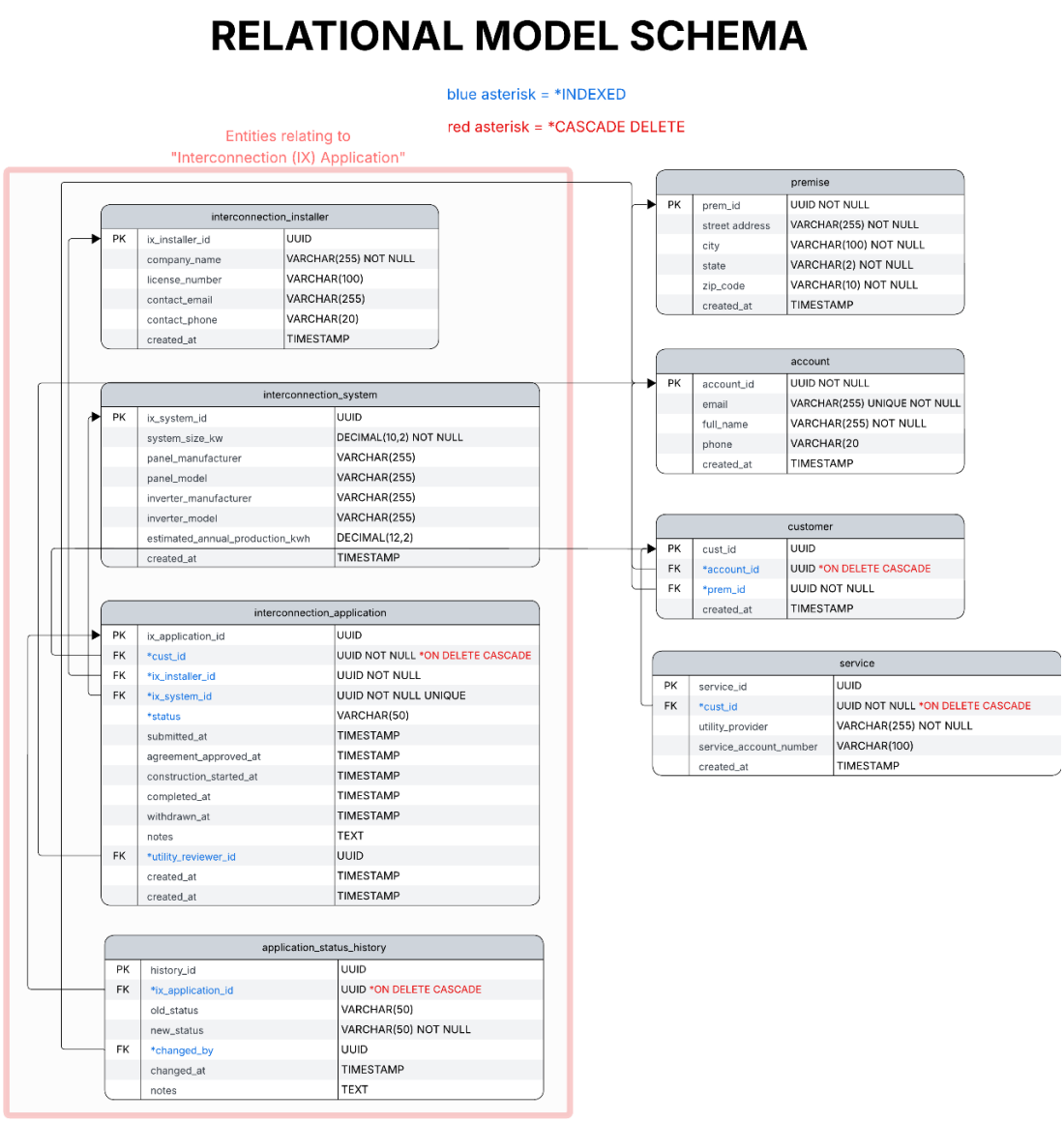


Figure A5: Full detailed schema used to create database, denoting indexed columns

Two very useful index columns are “status” and “prem_id”:

- interconnection_application.status
- customer.prem_id

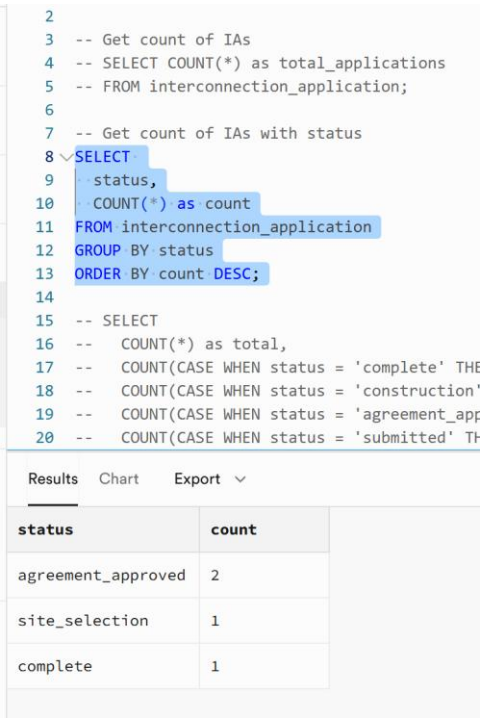
Figure A6: Entity Relationship Summary: Interconnection Portal Database

Relationship #	From Entity → To Entity	Cardinality	Foreign Key	Business Logic
1	account → customer	(0..*) to (1..1)	customer.account_id	An account can be associated with zero or many customers (e.g., multiple family members). Each customer must be assigned to exactly one account for login and contact information.
2	customer → premise	(0..1) to (0..*)	customer.prem_id	A customer may optionally reside at one premise (physical location). Multiple customers can share the same premise (e.g., roommates, multi-unit buildings). The relationship is optional because a customer can exist before a premise is assigned.
3	customer → service	(0..*) to (1..1)	service.cust_id	A customer can activate zero or many utility services (e.g., electricity, gas). Each service must belong to exactly one customer who is responsible for billing.
4	customer → interconnection_application	(0..*) to (1..1)	interconnection_application.cust_id	A customer can submit zero or many interconnection applications over time (e.g., multiple properties, system upgrades). Each application must be submitted by exactly one customer who is the legal applicant.
5	interconnection_installer → interconnection_application	(0..*) to (1..1)	interconnection_application.ix_installer_id	An installer company can be hired for zero or many applications. Each application must specify exactly one installer who is responsible for the solar system installation and commissioning.
6	interconnection_system → interconnection_application	(0..1) to (1..1)	interconnection_application.ix_system_id (UNIQUE)	**A solar system can be associated with at most one application (enforced by UNIQUE constraint).** Each application must specify exactly one solar system with technical specifications. The 1:1 relationship prevents duplicate applications for the same physical system.
7	account → interconnection_application (reviewer)	(0..*) to (0..1)	interconnection_application.utility_reviewer_id	A utility account (employee) can review zero or many applications. Each application can optionally be assigned to one utility reviewer for technical evaluation. This is optional because applications in early stages may not yet have an assigned reviewer.
8	interconnection_application → application_status_history	(0..*) to (0..1)	application_status_history.ix_application_id	An application can generate zero or many history records tracking status changes over time (e.g., submitted → approved → complete). Each history entry references at most one application. This creates an audit trail of workflow progression.
9	account → application_status_history (audit)	(0..*) to (0..1)	application_status_history.changed_by	An account can be recorded as the modifier for zero or many status changes (audit trail). Each history entry can optionally record which account made the change. This is optional because some automated system changes may not have a specific user associated with them.

Figure A6: Listing 9 key relationships and business logic. Note: Cardinality notation uses (min..max) format where 0 = optional, 1 = mandatory, * = many.

For example, (0..*) means "zero or many" and (1..1) means "exactly one (mandatory)".

Figure A7: Simple example backend SQL query to count applications by status



```
-- Get count of IAs
-- SELECT COUNT(*) as total_applications
-- FROM interconnection_application;

-- Get count of IAs with status
SELECT
  status,
  COUNT(*) as count
FROM interconnection_application
GROUP BY status
ORDER BY count DESC;

-- SELECT
--   COUNT(*) as total,
--   COUNT(CASE WHEN status = 'complete' THE
--   COUNT(CASE WHEN status = 'construction'
--   COUNT(CASE WHEN status = 'agreement_app
--   COUNT(CASE WHEN status = 'submitted' TH
```

Results | Chart | Export

status	count
agreement_approved	2
site_selection	1
complete	1

Figure A7: Example of using backend SQL query to show count of applications by status. The same information is available through front-end interface, but using the query is much more effective when dealing with larger volume applications (i.e. hundreds of apps).

Same view from the front-end shows 4 applications

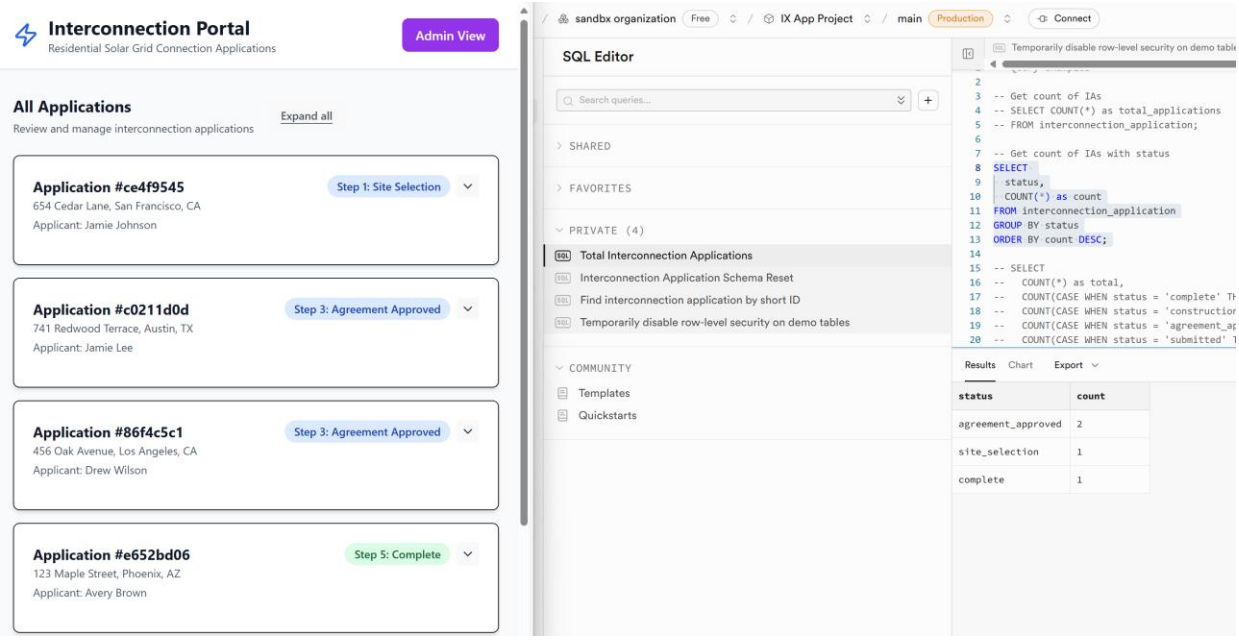


Figure A8: More powerful query, using JOIN LATERAL

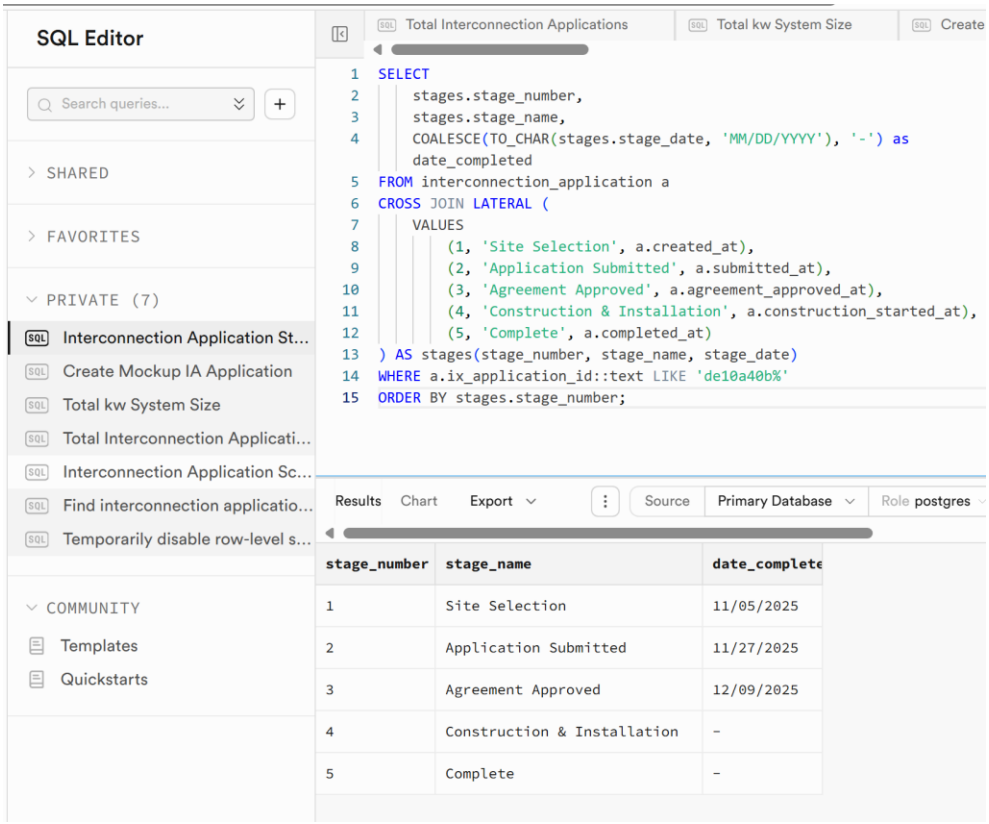


Figure A8: Highlighting an example of how JOIN LATERAL can be used on an individual application to list all the associated completion dates for each stage.

Same view front end shows application completed 3 stages

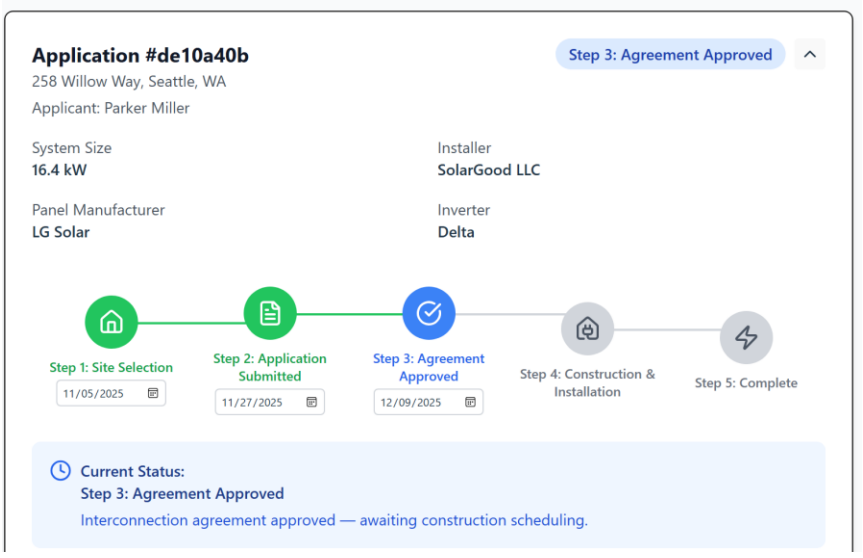


Figure A9: Query using multiple JOINs – view applications by solar installer

IX App Project / main Production Connect

Applications for Specific Installer | Total Interconnection Applications | Total kw System Size | Create Mo

```sql
-- Example of only showing applications assigned to a specific installer
SELECT
    i.company_name,
    SUBSTRING(a.ix_application_id::text, 1, 8) as app_id,
    a.status,
    acc.full_name as customer_name,
    p.street_address || ', ' || p.city || ', ' || p.state as location,
    s.system_size_kw,
    s.panel_manufacturer,
    TO_CHAR(a.created_at, 'MM/DD/YYYY') as submitted_date,
    TO_CHAR(a.completed_at, 'MM/DD/YYYY') as completion_date
FROM interconnection_application a
JOIN interconnection_installer i ON a.ix_installer_id = i.ix_installer_id
JOIN customer c ON a.cust_id = c.cust_id
JOIN account acc ON c.account_id = acc.account_id
LEFT JOIN premise p ON c.prem_id = p.prem_id
JOIN interconnection_system s ON a.ix_system_id = s.ix_system_id
WHERE i.company_name = 'SuperSolar LLC'  -- Replace with actual installer name
ORDER BY a.created_at DESC;
```

Results | Chart | Export

company_name	app_id	status	customer_name	location	system_size_k
SuperSolar LLC	6692e1e8	site_selection	Drew Thomas	456 Oak Avenue, Portland, OR	39.50
SuperSolar LLC	59b037f2	agreement_approved	Quinn Garcia	741 Redwood Terrace, Atlanta, GA	2.30

Figure A9: Another query example showing multiple JOINs used to list all applications belonging to a single solar installer (i.e. company name = SuperSolar LLC)

The same 2 applications viewed from the front-end show both belong to “Super Solar LLC”