Big3 Construction Company
Database Design & Normalization Report

Database Normalization: 1NF through 5NF

Team Members Levis Ishimwe (i.levis@alustudent.com)
Obasi-otani Owai Ibe (o.ibe@alustudent.com), Fatoumata Ndiaye(f.ndiaye@alustudent.com)
Submission Date
February 19, 2025

Original Data Overview

Big3 Construction Company has been managing its project data in a single flat spreadsheet with 26 columns and 16
rows. The table below summarises the key columns and the normalization problems each creates.

Column(s)	Problem
WorkerSkills	Multi-valued — pipe-separated (e.g. Carpentry|Framing)
WorkerCertifications	Multi-valued — pipe-separated (e.g. OSHA|First Aid)
SupplierPhones	Multi-valued — pipe-separated (e.g. 617-555-9000|617-555-9001)
MaterialSupplied / MaterialUnitCost	Multi-valued pairs — pipe-separated
EquipmentUsed / EquipmentRentalCost	Multi-valued pairs — pipe-separated
ProjectName, StartDate, EndDate...	Repeated for every worker row of the same project
ClientName, ClientPhone, ClientEmail	Repeated on every row belonging to the same client
SupervisorName, SupervisorPhone	Repeated on every row of the same project
ClientCity	Depends on ClientName, not on ProjectID (transitive dependency)

First Normal Form (1NF)

1. Problem Identification

1NF Rule: Every column must hold a single, atomic (indivisible) value. There must be no repeating groups and each
row must be uniquely identifiable.

Violations found in the original spreadsheet:
• WorkerSkills contains pipe-separated values: Carpentry|Framing
• WorkerCertifications: OSHA|First Aid
• SupplierPhones: 617-555-9000|617-555-9001
• MaterialSupplied: Concrete|Steel paired with MaterialUnitCost: 120|300
• EquipmentUsed: Crane|Bulldozer paired with EquipmentRentalCost: 5000|3000
• Multiple rows for the same project (P001 appears 3 times) act as a repeating group.

2. Transformation — 1NF Table Structure

Each pipe-separated value is split into its own row. Multi-valued columns are removed from the main table and placed
in dedicated tables. The result is a flat table where every cell contains exactly one value.

1NF: project_data (flat, atomic)

Column	Data Type	Notes
project_id	VARCHAR(10)	PK
project_name	VARCHAR(100)	
project_type	VARCHAR(50)	
start_date	DATE	
end_date	DATE	
site_address	VARCHAR(200)	
site_city	VARCHAR(50	
site_state	CHAR(2)	
client_name	VARCHAR(100)	
client_phone	VARCHAR(20)	
client_email	VARCHAR(100)	
client_city	VARCHAR(50)	
supervisor_name	VARCHAR(100)	
supervisor_phone	VARCHAR(20)	
worker_name	VARCHAR(100	Part of composite key
worker_phone	VARCHAR(20)	
skill	VARCHAR(100)	Atomic — one skill per row
certification	VARCHAR(100)	Atomic — one cert per row

hourly_rate	DECIMAL(8,2)	
supplier_name	VARCHAR(100)	
supplier_city	VARCHAR(50)	
supplier_phone	VARCHAR(20)	Atomic — one phone per row
material_name	VARCHAR(100)	Atomic — one material per row
material_unit_cost	DECIMAL(10,2)	
equipment_name	VARCHAR(100)	Atomic — one piece per row
equipment_rental_cost	DECIMAL(10,2)	

3. Justification

Splitting pipe-separated values means every cell now holds a single value. SQL WHERE clauses, GROUP BY, and
aggregate functions can now be applied to skills, certifications, phones, materials, and equipment directly. Data entry
errors like mismatched pair counts (3 materials but only 2 costs) are also prevented.

Second Normal Form (2NF)

1. Problem Identification 2NF Rule:

Must be in 1NF, and every non-key attribute must depend on the whole primary key — no partial dependencies.

In the 1NF table the composite key is (project_id, worker_name, skill, certification, supplier_name, supplier_phone, material_name, equipment_name). Many columns depend only on part of this key:
• project_id alone → project_name, project_type, start_date, end_date, site_address, site_city, site_state
• project_id alone → client_name, client_phone, client_email, client_city (via client)
• project_id alone → supervisor_name, supervisor_phone
• worker_name alone → worker_phone, hourly_rate
• supplier_name alone → supplier_city
• material_name alone → material_unit_cost (mostly)

2. Transformation 2NF Table Structures

Partial dependencies are removed by extracting each independently determined entity into its own table

Column	Data Type	Notes
project_id	VARCHAR(10)	
project_name	VARCHAR(100)	
project_type	VARCHAR(50)	
start_date	DATE	
end_date	DATE	
site_address	VARCHAR(200)	
site_city	VARCHAR(50	
site_state	CHAR(2)	
client_name (FK)	VARCHAR(100)	client
Supervisor_name (FK)	VARCHAR(100)	supervisors

Table: workers

Column	Data Type	Notes
Worker_name (PK)	VARCHAR(100)	
worker_phone	VARCHAR(20)	
hourly_rate	DECIMAL(8,2)	

Table: suppliers

Column	Data Type	Notes
supplier_name	VARCHAR(100)	
supplier_city	VARCHAR(50)	

3. Justification

Extracting projects, workers, and suppliers into their own tables means project information is stored exactly once regardless of how many workers are assigned. Updating a project's end date now requires changing a single row instead of every row that shared that project in 1NF.

Third Normal Form (3NF)

1. Problem Identification
3NF Rule: Must be in 2NF, and no non-key attribute may depend on another non-key attribute (no transitive dependencies).
Transitive dependencies found after 2NF:
• In **projects:** client_name → client_phone, client_email, client_city (client details depend on client_name, not project_id)
• In **projects**: supervisor_name → supervisor_phone (supervisor phone depends on supervisor_name, not project_id)

2. Transformation — 3NF Table Structures
Table: clients

Column	Data Type	Notes
client_id (PK)	INT AUTO_INCREMENT	
client_name	VARCHAR(100)	UNIQUE NOT NULL
client_phone	VARCHAR(20)	
client_email	VARCHAR(100)	
client_city	VARCHAR(50)	

Table: supervisors

Column	Data Type	Notes
Supervisor_id (PK)	INT AUTO_INCREMENT	
supervisor_name	VARCHAR(100)	UNIQUE NOT NULL
supervisor_phone	VARCHAR(20)	

Table: projects (updated)

Column	Data Type	Notes
Project_id (PK)	VARCHAR(10)	
project_name	VARCHAR(100)	
project_type	VARCHAR(50)	
start_date	DATE	
end_date	DATE	
site_address	VARCHAR(200)	
site_city	VARCHAR(50)	
site_state	CHAR(2)	
Client-id (FK)	INT	Clients (client_id)
Supervisor_id (FK)	INT	supervisors(superviser_id)

3. Justification

Client and supervisor data now live in their own tables referenced by surrogate integer keys. If Metro Corp changes their phone number, one UPDATE on the clients table propagates everywhere. Previously this would have required updating every project row associated with Metro Corp.

Boyce-Codd Normal Form (BCNF)

BCNF Rule: For every functional dependency X → Y, X must be a superkey (a candidate key or a superset of one).
After 3NF we verify all functional dependencies. In the current design:
• In clients: client_id → all other columns (client_id is PK)
• In projects: project_id → all other columns
• In workers: worker_id → all other columns
• In suppliers: supplier_id → all other columns
• No non-superkey determinants exist, the design already satisfies BCNF

2. Transformation

No structural changes are required. The tables created during 3NF already satisfy BCNF because every determinant in every table is a candidate key.

3. Justification

BCNF is a stronger version of 3NF. Because we used surrogate primary keys (client_id, supervisor_id, worker_id, supplier_id) throughout, there are no overlapping candidate keys that could create BCNF violations. The design is confirmed BCNF-compliant.

Fourth Normal Form (4NF)

4NF Rule: A table must be in BCNF and must not contain more than one independent multi-valued fact about an
entity.
Even in **BCNF,** a table can store two independent multi-valued facts, creating spurious combinations. In our design
after **BCNF:**

• Workers have multiple skills AND multiple certifications independently. A worker's skills have no dependency on their certifications — they are separate facts.
• Storing both in one table (worker_id, skill, certification) would force every (skill, certification) combination to be
stored, creating redundancy.
• Similarly, suppliers have multiple phone numbers — an independent multi-valued fact that must be separated from supplier-material relationships.

2. Transformation — 4NF Table Structures

Each independent multi-valued fact gets its own table:

Table: worker_skills

Column	Data Type	Notes
worker_id	(PK, FK) INT	workers(worker_id)
skill	(PK)	VARCHAR(100)

Table: worker_certifications

Column	Data Type	Notes
worker_id (PK, FK)	INT	INT workers(worker_id)
certification (PK)	VARCHAR(100)	

Table: supplier_phones

Column	Data Type	Notes
supplier_id (PK, FK)	INT	suppliers(supplier_id)
phone_number (PK)	VARCHAR(20)	

3. Justification

Separating skills and certifications into distinct tables means adding a new skill to a worker does not require inserting
new certification rows, and vice versa. The tables contain only one multi-valued fact each, eliminating all 4NF

Violations.

Fifth Normal Form (5NF)

1. Problem Identification
5NF Rule: A table must be in 4NF and must not contain any join dependency that is not implied by its candidate keys.
In other words, a table cannot be losslessly decomposed further.
A ternary relationship exists between Project, Supplier, and Material. The data shows:
• P001 uses BuildPro Supplies for Concrete and Steel
• P001 also uses SteelWorks Inc for Steel Beams
• P007 uses SteelWorks for Steel Beams and Concrete, and Oceanic for Rebar
• Each (project, supplier, material) triple is an independent fact with its own unit_cost
• This ternary relationship cannot be decomposed into binary tables without losing data
2. Transformation — 5NF Table Structure

Table: project_supplier_materials

Column	Data Type	Notes
project_id (PK, FK)	VARCHAR(10)	projects(project_id)
supplier_id (PK, FK)	INT	suppliers(supplier_id)
material_id (PK, FK)	INT	materials(material_id)
unit_cost	DECIMAL(10,2)	Cost of this material from this supplier on this project

Table: materials

Column	Data Type	Notes
material_id (PK)	INT	AUTO_INCREMENT material_name VARCHAR(100)
material_name	VARCHAR(100)	UNIQUE NOT NULL

3. Justification
The ternary table project_supplier_materials captures the fact that a specific supplier provides a specific material on a
specific project at a given cost. Decomposing this into binary tables (project-supplier, project-material,
supplier-material) would cause incorrect data to be inferred on re-join — a classic 5NF violation. The ternary table preserves this fact correctly and cannot be further decomposed without losing information.
preserves this fact correctly and cannot be further decomposed without losing information.

Final 5NF Schema Summary

All 13 tables in the final normalised design:

Table	Primary Key	Foreign Keys	Purpose
clients	client_id	—	Stores client details
supervisors	supervisor_id	—	Stores supervisor details
projects	project_id	client_id, supervisor_id	One row per project
workers	worker_id	—	Stores worker details & pay
worker_skills	(worker_id, skill)	worker_id	4NF: independent skills
worker_certifications	(worker_id, certification)	worker_id	4NF: independent certs
project_workers	(project_id, worker_id)	project_id, worker_id	Assignment bridge table
suppliers	supplier_id	—	Stores supplier details
supplier_phones	(supplier_id, phone_number)	supplier_id	4NF: multiple phones
materials	material_id	—	Material catalogue
project_supplier_materials	(project_id, supplier_id, material_id)	project_id, supplier_id, material_id	5NF: ternary relationship
equipment	equipment_id	—	Equipment catalogue
project_equipment	(project_id, equipment_id)	project_id, equipment_id	Equipment per project

Entity Relationship Diagram (ERD)

This diagram below shows all entities and their relationships. PK = Primary Key, FK = Foreign Key.

clients ……. **projects** ……... **supervisors**

(client_id PK) (project_id PK) (supervisor_id PK)
client_name project_name supervisor_name
Client_phone project_type supervisor_phone
Client_email start_date
Client_city end_date
site_address
site_city, site_state
client_id FK ……
supervisor_id FK ……
……

……

…… …….. …….

project_workers project_supplier_materials project_equipment
(project_id FK, PK) project_equipment
(worker_id FK, PK) (supplier_id FK, PK) (equipment_id FK, PK)
(material_id FK, PK) rental_cost
unit_cost

…… …….. …….

workers suppliers equipment
(worker_id PK) (supplier_id PK) (equipment_id PK)
worker_name supplier_name equipment_name
worker_phone supplier_city
hourly_rate

…… ……..

worker_skills worker_certifications supplier_phones
(worker_id FK, PK) (worker_id FK, PK) (supplier_id FK, PK)
skill certification phone_number
materials
(material_id PK)
material_name
(referenced by project_supplier_materials)

Team Member Contributions

Member	Email	Contributions
Levis Ishimwe	i.levis@alustudent.com	1NF & 2NF Documentation, SQL CREATE TABLE, SQL INSERT, README

Obasi-otani Owai Ibe	o.ibe@alustudent.com	3NF, BCNF, 4NF & Documentation
Fatoumata Ndiaye	f.ndiaye@alustudent.com	5NF Documentation, ERD Design,Verification Queries, Report Compilation