

Databases

Normalisation

8 February 2024

克明峻德，格物致知

Quick Recap: Entity-Relationship Diagram

Key Concepts:


- Functional dependencies ensure data consistency and integrity, crucial for preventing update anomalies due to inconsistent updates. Closures aid in identifying all attributes determined by a set, key in avoiding insertion anomalies by ensuring completeness in data records..
- Superkeys and keys, derived through closures, play a significant role in structuring databases to minimize redundancy and potential anomalies. Primary keys, a type of key, are specifically vital in averting deletion anomalies by ensuring each record is uniquely identifiable.

Customer's Order Example

OrderID	ProductID	Quantity
0100	P001	2
0101	P002	1
0100	P003	1

- Functional Dependencies and Anomaly Analysis

Attribute Set	Functional Dependency	Anomaly Detection
(OrderID, ProductID)	(OrderID, ProductID) → Quantity	No update anomaly expected



Imagine a grocery store with products placed randomly, leading to a chaotic shopping environment.

This disorganization causes shopper frustration and complicates inventory management.

Q: How can reorganizing the store into logical sections streamline shopping and inventory management?

A: Just as organizing store aisles improves product accessibility and management, database normalization arranges data into structured tables, reducing redundancy and enhancing efficiency.

Crafting a Good Data Model

What Makes a Good Data Model?

- **Minimal Modification Anomalies:** A robust model ensures that data remains consistent and accurate through all insertions, updates, and deletions.
- **Efficient Data Usage:** Optimizes data storage and retrieval, preventing unnecessary duplication.
- **Flexibility and Scalability:** Adapts easily to evolving data requirements and scales with growing data volumes.

What Must Be Done: Reduce Redundancies

- **Identify and Eliminate Duplication:** Reducing redundancies in a database prevents data inconsistencies and lowers storage needs. It involves ensuring each piece of information exists only once, eliminating duplicates.
- **Streamline Data Relationships:** By organizing data to ensure that each item is stored in the most logical and efficient way, databases are structured for better accessibility and clearer understanding.

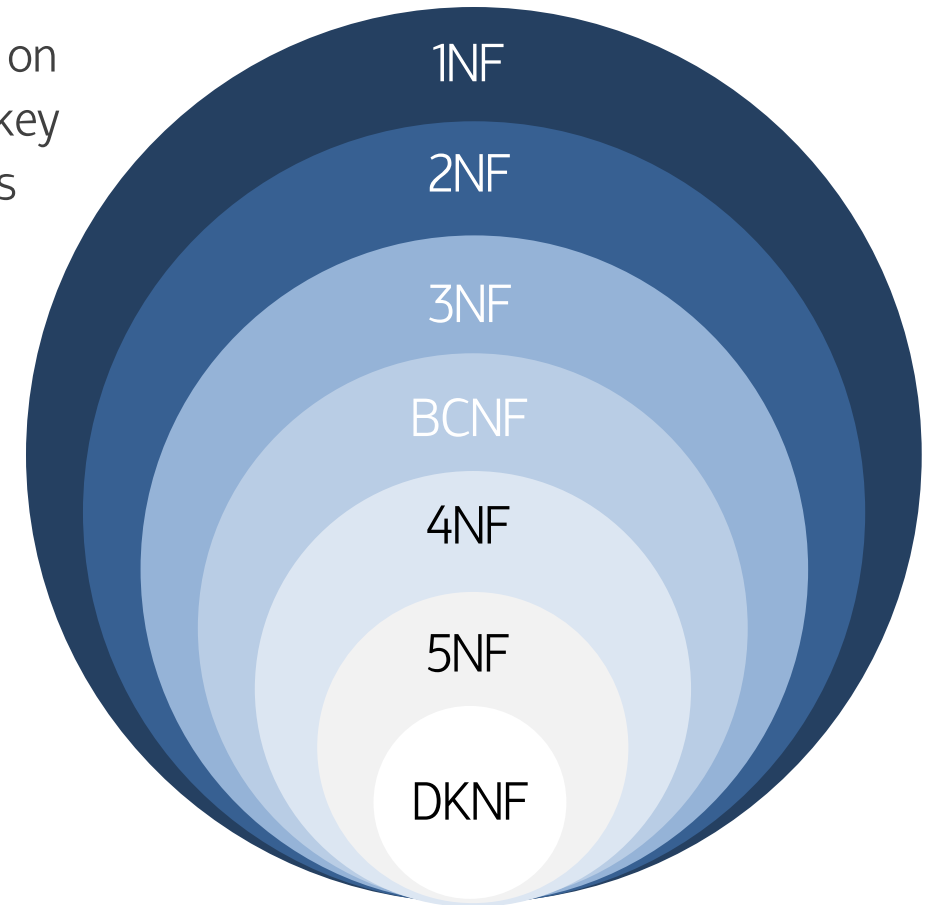
Normalization: The How-To Process

- **Systematic Organization:** Normalization is a process to organize data in databases, minimizing redundancy and anomalies through structured stages, or 'normal forms'.
- **Staged Refinement:** The process progresses through stages from First Normal Form (1NF) to more advanced forms like BCNF, each stage reducing data duplication and enhancing integrity and efficiency.
- **Practical Balance:** Normalization balances data integrity with database performance, tailored to specific needs and scenarios.

Normal Forms

Key Points

- **Cumulative Nature:** Each normal form adds rules to further refine the database structure. Achieving a higher normal form (like BCNF) implies compliance with all lower forms' criteria.
- **Practical Usage:** In current industry practices, emphasis is often on achieving BCNF due to its effectiveness and clarity in resolving key anomalies. 2NF, while historically significant, is less a focus as its requirements are naturally met on the path to 3NF and BCNF.

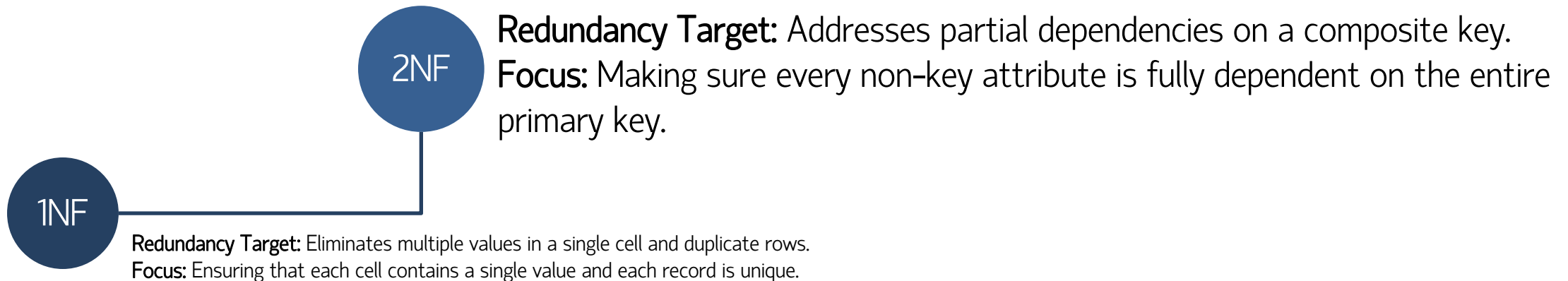


Staged Refinement

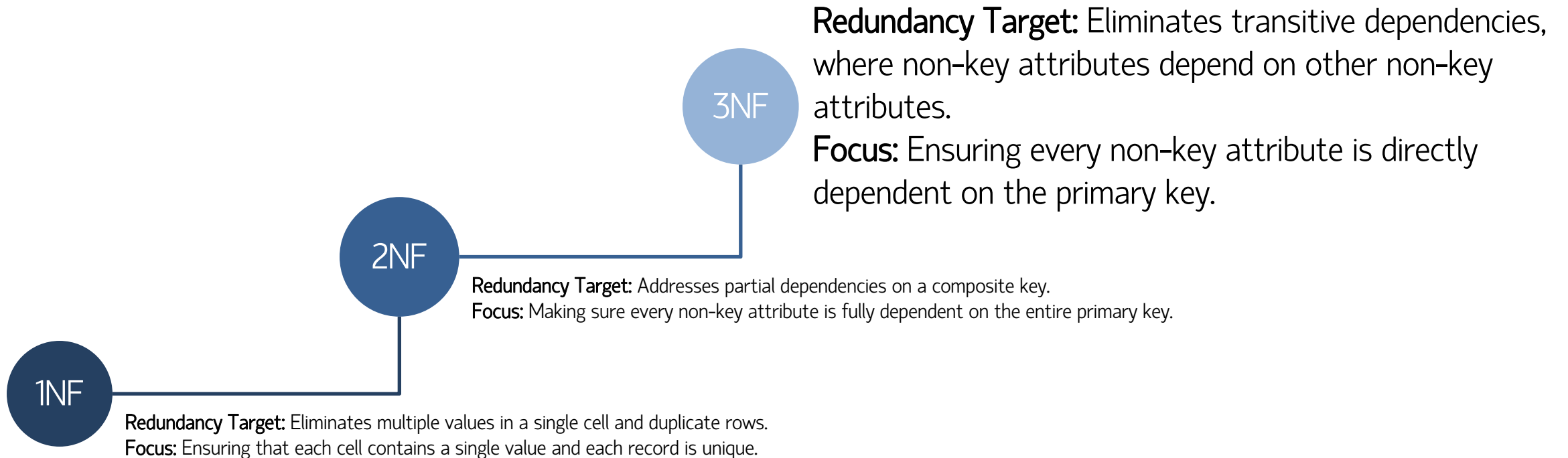


Redundancy Target: Eliminates multiple values in a single cell and duplicate rows.
Focus: Ensuring that each cell contains a single value and each record is unique.

Staged Refinement



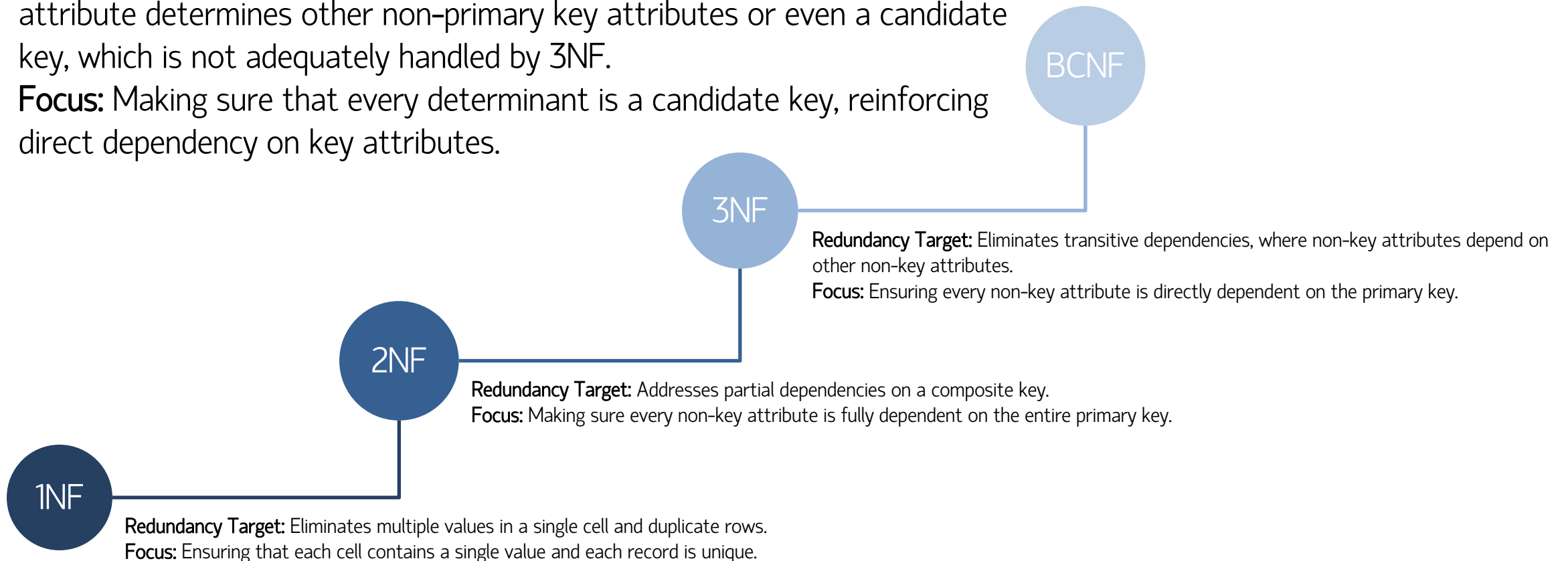
Staged Refinement



Staged Refinement

Redundancy Target: Specifically addresses cases where a non-primary key attribute determines other non-primary key attributes or even a candidate key, which is not adequately handled by 3NF.

Focus: Making sure that every determinant is a candidate key, reinforcing direct dependency on key attributes.



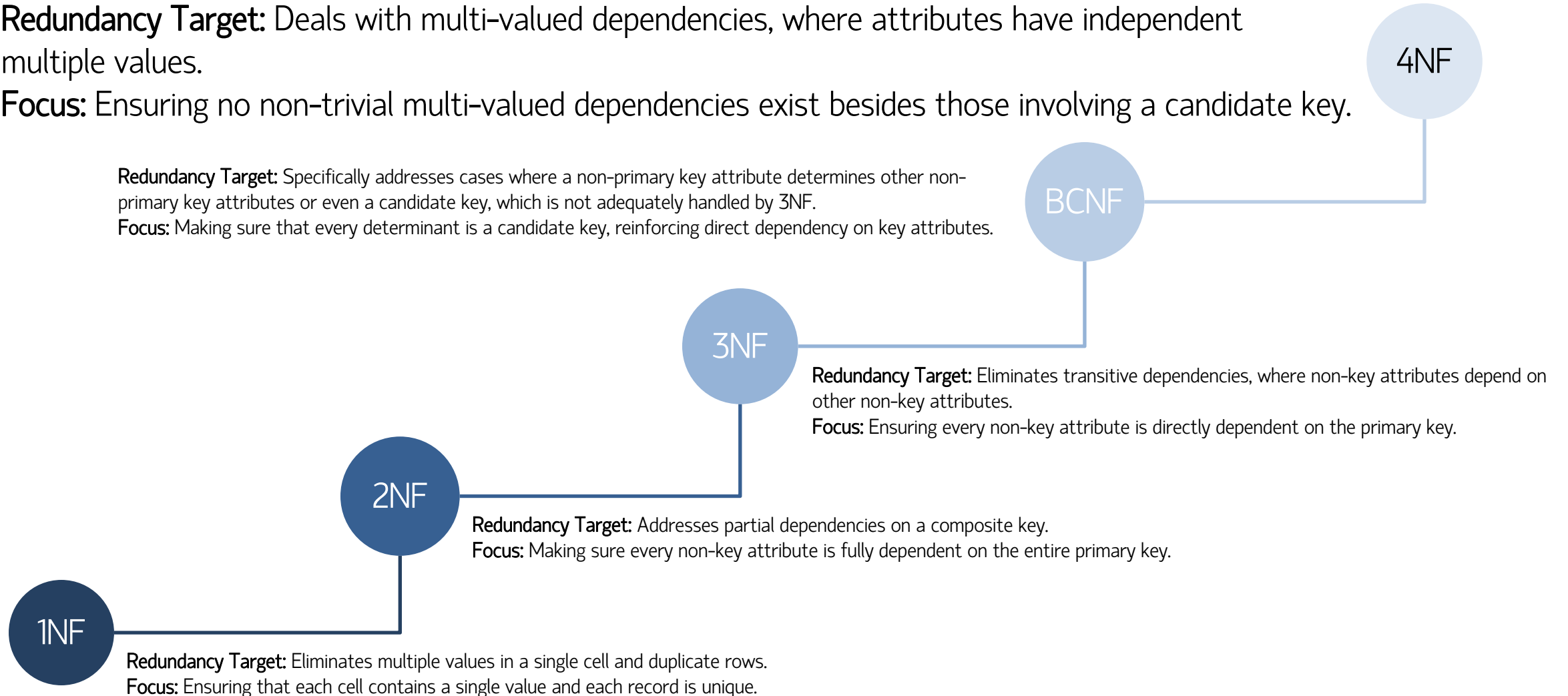
Staged Refinement

Redundancy Target: Deals with multi-valued dependencies, where attributes have independent multiple values.

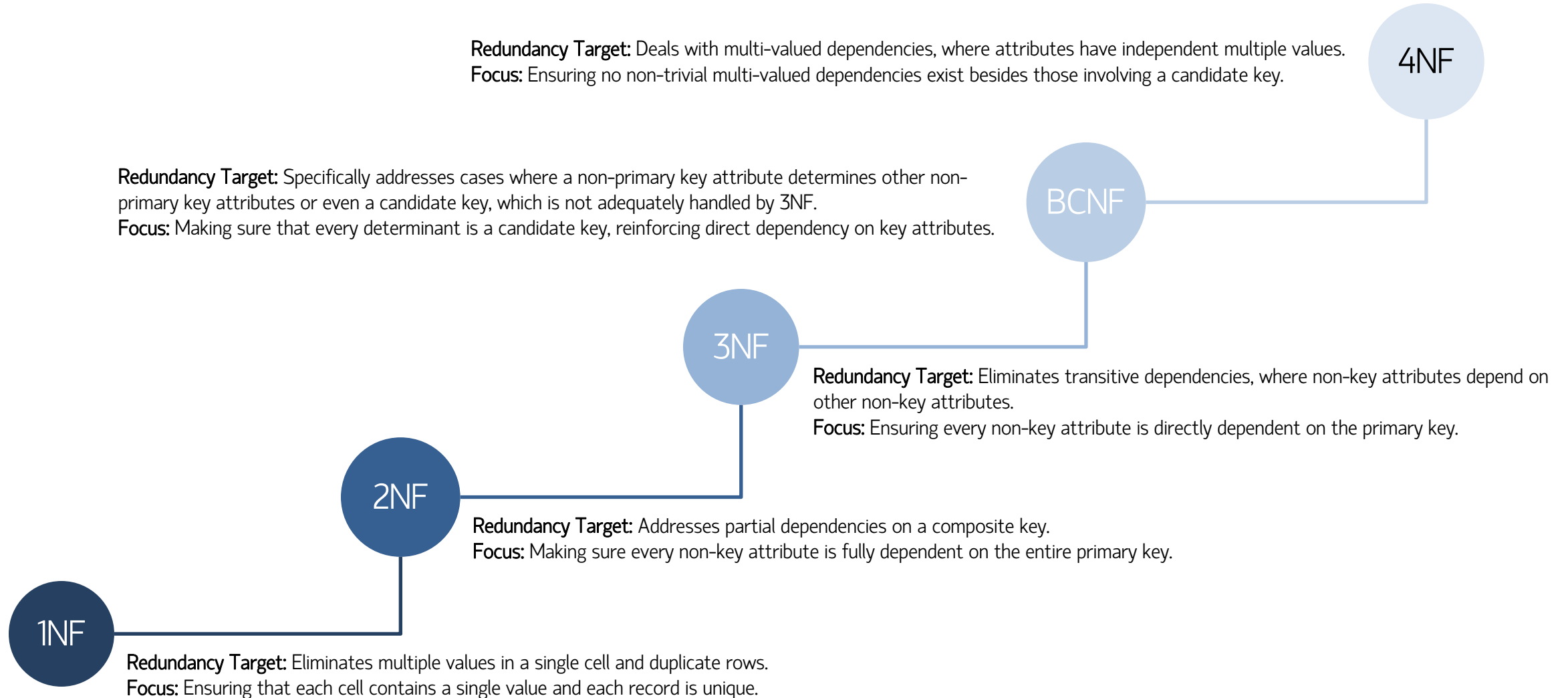
Focus: Ensuring no non-trivial multi-valued dependencies exist besides those involving a candidate key.

Redundancy Target: Specifically addresses cases where a non-primary key attribute determines other non-primary key attributes or even a candidate key, which is not adequately handled by 3NF.

Focus: Making sure that every determinant is a candidate key, reinforcing direct dependency on key attributes.



Staged Refinement



Staged Refinement



Redundancy Target: Eliminates multiple values in a single cell and duplicate rows.
Focus: Ensuring that each cell contains a single value and each record is unique.

Understanding First Normal Form (1NF)

What is 1NF?

- 1NF requires that each cell in a table has a unique, atomic value and that each record is distinct. It's the first step in ensuring data is well-structured and unambiguous

Spotting the Problem

- Look for columns containing multiple values. In our example, 'Courses' and 'Textbooks' hold lists, violating the 1NF rule.

Sample Data (Not in 1NF)

StudentID	Courses	Textbooks
101	Math, Science	Math101, Science201
102	Literature, Math	Lit101, Math101

Breaking Down Complexity

- 1NF aims to simplify database records. Instead of storing multiple values in a single cell, 1NF requires splitting these into individual, atomic items. This reduces complexity and enhances data clarity.

Transitioning to First Normal Form (1NF)

Normalisation

- To meet 1NF, multi-valued fields are split into separate rows, creating a one-to-one relationship between key and value in each cell.
- This leads to a more organized table where each data point is distinctly represented in its own row, complying with the atomicity principle of 1NF.

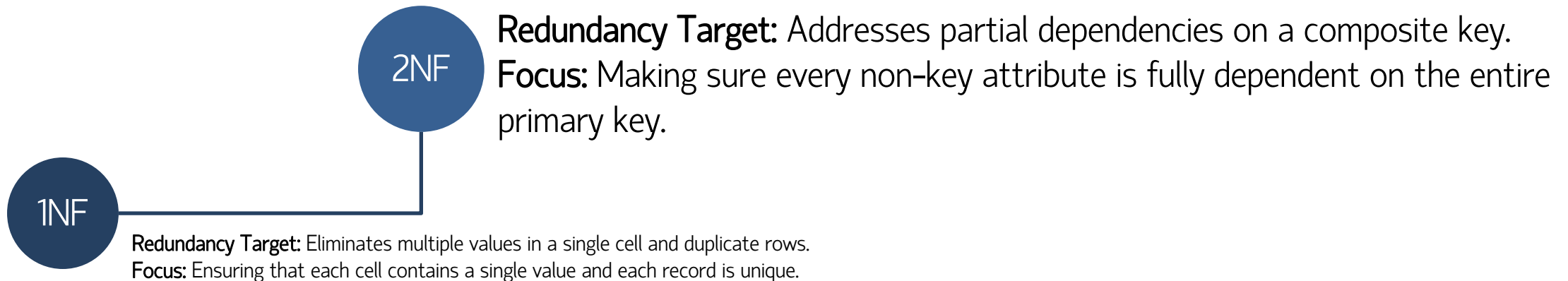
Transformed into 1NF

StudentID	Course	Textbook
101	Math	Math101
101	Science	Science201
102	Literature	Lit101
102	Math	Math101

Understanding the Change

- The table now clearly represents each piece of information in separate, atomic units, laying the groundwork for more advanced data structuring and integrity.

Staged Refinement



Understanding Second Normal Form (2NF)

What is 2NF?

- Second Normal Form (2NF) builds upon 1NF by addressing partial dependencies. In 2NF, every non-key attribute must fully depend on the entire primary key, not just part of a composite key.

Sample Data (1NF but not in 2NF)

StudentID	Course	Textbook
101	Math	Math101
101	Science	Science201
102	Literature	Lit101
102	Math	Math101

Eliminating Partial Dependencies

- In 2NF, we target the elimination of these partial dependencies. In our example, if 'Textbook' is related to 'Course' but not to the combination of 'StudentID' and 'Course', it represents a partial dependency.

Functional Dependencies and Keys

Functional Dependencies

- We analyze how attributes determine each other in our table. For example:

```
StudentID -> Course, Textbook
Course -> Textbook
```

- This analysis shows that 'Course' and 'Textbook' are functionally dependent on 'StudentID'. However, 'Textbook' also depends directly on 'Course', indicating a partial dependency.

Closure and Keys Analysis

- Closure helps us understand what attributes can be determined from a given set and whether it forms a key. Let's analyze some attribute sets from our data:

Attribute Set	Closure	Is Superkey?	Is Key?	Anomaly Indication
StudentID	StudentID, Course, Textbook	No	No	Partial Dependency Anomaly (Textbook depends only on Course)
Course	Course, Textbook	No	No	No apparent anomaly
StudentID, Course	StudentID, Course, Textbook	Yes	Yes (Candidate Key)	No anomaly

- This analysis reveals that combining 'StudentID' and 'Course' forms a candidate key without any anomaly, aligning with 2NF requirements.

Transitioning to Second Normal Form (2NF)

Normalisation

- The goal is to restructure data so that all information in the table is fully dependent on the complete primary key.
- In our 1NF table, the 'Textbook' is dependent only on 'Course', not the composite key of 'StudentID' and 'Course'. This is a partial dependency, which 2NF aims to resolve.

Transformed into 2NF

- StudentCourses

StudentID	Course
101	Math
101	Science
102	Literature
102	Math

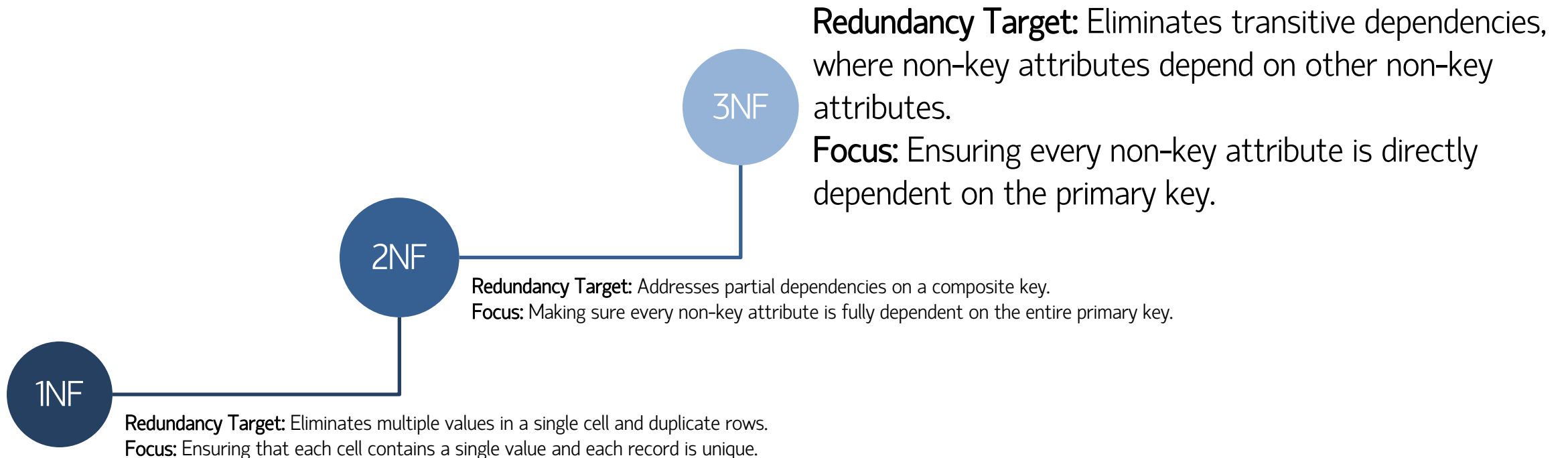
- CourseDetails

Course	Textbook
Math	Math101
Science	Science201
Literature	Lit101

Understanding the Change

- By dividing the original table into 'StudentCourses' and 'CourseTextbooks', we have removed partial dependencies, aligning the database with the principles of 2NF.

Staged Refinement



Understanding Third Normal Form (3NF)

What is 3NF?

- Third Normal Form (3NF) builds upon the foundation of 2NF by eliminating transitive dependencies. In 3NF, every non-key attribute must be directly dependent on the primary key, not through another attribute.

Sample Data (2NF but not in 3NF)

- EmployeeDepartment

EmployeeID	EmployeeName	Department	DepartmentLocation
E001	Alice	HR	New York
E002	Bob	Marketing	Chicago
E003	Charlie	HR	New York
E004	Dana	Engineering	San Francisco

Eliminating Transitive Dependencies

- In 3NF, we target the elimination of these transitive dependencies. For instance, 'DepartmentLocation' is dependent on 'Department', which in turn is dependent on 'EmployeeID'.

Functional Dependencies and Closure

Functional Dependencies

- In the initial EmployeeDetails table, we identify the following dependencies:

```
EmployeeID -> EmployeeName, Department, DepartmentLocation
Department -> DepartmentLocation
```

- The first dependency shows each employee ID determining name, department, and department location. The second highlights a transitive dependency: the department determines its location.

Closure and Keys Analysis

- Closure analysis helps identify the full set of attributes determined by a key:

Attribute Set	Closure	Is Superkey?	Is Key?	Transitive Dependency?
EmployeeID	EmployeeID, EmployeeName, Department, DepartmentLocation	Yes	Yes	Yes, DepartmentLocation is transitively dependent on EmployeeID via Department
Department	Department, DepartmentLocation	No	No	Yes, shows transitive dependency

- The analysis reveals that 'DepartmentLocation' is transitively dependent on 'EmployeeID' via 'Department', a clear indicator for the need to transition to 3NF.

Transitioning to Third Normal Form (3NF)

Normalisation

- The goal is to eliminate transitive dependencies in a database table, ensuring that all non-key attributes are not only fully functionally dependent on the primary key but also non-transitively dependent.
- In our 2NF table, 'DepartmentLocation' should not depend on 'EmployeeID' through 'Department'. To align with 3NF, we need to remove this transitive dependency by restructuring the data into separate tables.

Transition into 3NF

- EmployeeDepartment

EmployeeID	EmployeeName	Department
E001	Alice	HR
E002	Bob	Marketing
E003	Charlie	HR
E004	Dana	Engineering

- DepartmentLocations

Department	DepartmentLocation
HR	New York
Marketing	Chicago
Engineering	San Francisco

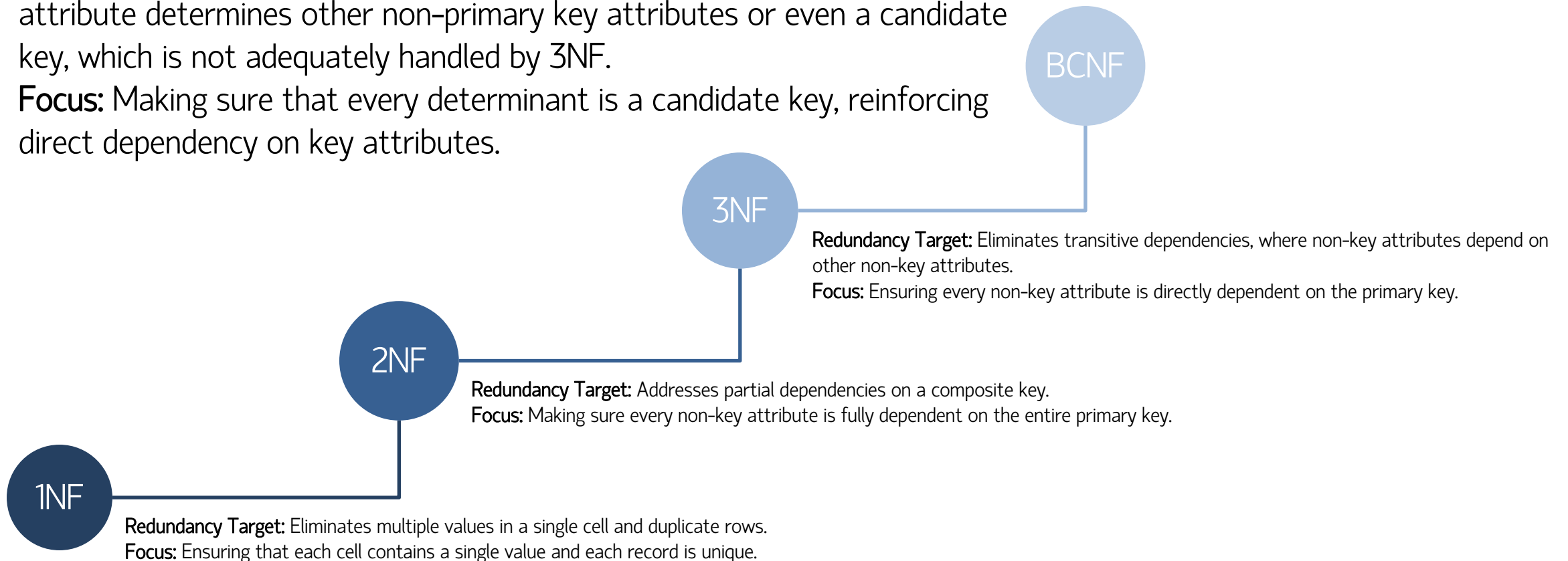
Understanding the Change

- In the EmployeeDepartment table, each employee's details directly depend on their unique EmployeeID, exemplifying 3NF's requirement for direct dependency between the primary key and non-key attributes.
- The separation into EmployeeDepartment and DepartmentLocations tables resolves transitive dependencies. Department locations are now independently determined.

Staged Refinement

Redundancy Target: Specifically addresses cases where a non-primary key attribute determines other non-primary key attributes or even a candidate key, which is not adequately handled by 3NF.

Focus: Making sure that every determinant is a candidate key, reinforcing direct dependency on key attributes.



Understanding Boyce-Codd Normal Form (BCNF)

What is BCNF?

- A relation is in Boyce-Codd Normal Form (BCNF) if, and only if, every determinant in a functional dependency is a candidate key.
- This prevents any situation where a non-candidate-key column determines another column, a scenario that can lead to anomalies.

Sample Data (Not in BCNF)

- EquipmentAllocation

EquipmentID	EmployeeID	EmployeeDept
EQ101	E001	HR
EQ102	E002	Marketing
EQ103	E001	HR

Eliminating Transitive Dependencies

- In BCNF, the focus is on removing dependencies where a non-superkey determines another attribute. For example, in the original EquipmentAllocation table, EmployeeID determined EmployeeDept but wasn't a superkey. BCNF demands restructuring to ensure that all determinants are candidate keys.

Functional Dependencies and Closure

Functional Dependencies

- The initial BookDetails table presents several dependencies:

```
EquipmentID -> EmployeeID, EmployeeDept (In EquipmentAllocation table)
EmployeeID -> EmployeeDept
```

- BCNF requires that for each dependency, the determinant should be a superkey. 'EmployeeID' determines 'EmployeeDept', but 'EmployeeID' is not a superkey in the EquipmentAllocation table.

Closure and Keys Analysis

- Using closure, we assess which attributes can be fully determined by a given set of attributes. This analysis helps us identify superkeys and understand BCNF compliance:

Attribute Set	Closure	Is Superkey?	Is Key?	BCNF Compliance
EquipmentID	EquipmentID, EmployeeID, EmployeeDept	Yes	Yes	Compliant (Determinant is a superkey)
EmployeeID	EmployeeID, EmployeeDept	No	No	Not Compliant (Determinant not a superkey)

- While 'EquipmentID' is a superkey in its table, 'EmployeeID' in the EquipmentAllocation table is not a superkey. This violates BCNF as 'EmployeeID' determines 'EmployeeDept'.

Transitioning to Boyce-Codd Normal Form (BCNF)

Normalisation

- Our focus is to ensure that every determinant is a candidate key, reinforcing direct dependency on key attributes.
- To adhere to BCNF, we must decompose the EquipmentAllocation table, separating the 'EmployeeDept' information into its own structure.

Transition into BCNF

■ EquipmentAssignment

EquipmentID	EmployeeID
EQ101	E001
EQ102	E002
EQ103	E001

■ EmployeeDepartment

EmployeeID	EmployeeDept
E001	HR
E002	Marketing

Understanding the Change

- We split 'EquipmentAllocation' into 'EquipmentAssignment' (links equipment to employees) and 'EmployeeDepartment' (maps employees to departments). This separation addresses the BCNF issue by ensuring 'EmployeeID' is a superkey in 'EmployeeDepartment'.
- The restructuring led to clearer and more efficient management of equipment and departmental data. Each table now has a singular focus, reducing redundancy and potential anomalies,

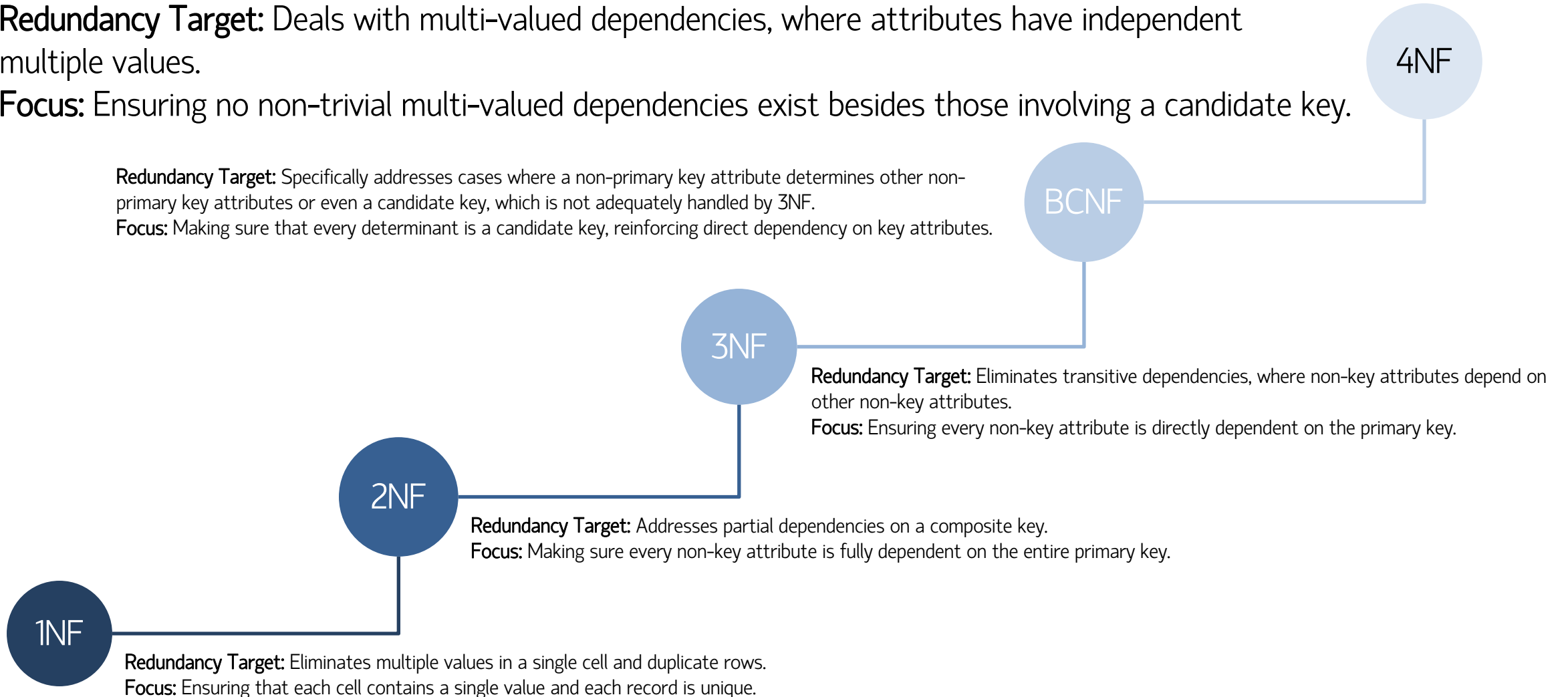
Staged Refinement

Redundancy Target: Deals with multi-valued dependencies, where attributes have independent multiple values.

Focus: Ensuring no non-trivial multi-valued dependencies exist besides those involving a candidate key.

Redundancy Target: Specifically addresses cases where a non-primary key attribute determines other non-primary key attributes or even a candidate key, which is not adequately handled by 3NF.

Focus: Making sure that every determinant is a candidate key, reinforcing direct dependency on key attributes.



Understand Fourth Normal Form (4NF)

What is 4NF?

- A table is in 4NF if, for every one of its non-trivial multi-valued dependencies, each attribute is independent of the others, except for the primary key.

Sample Data (Not in 4NF)

- 'EmployeeTrainingProjects'

EmployeeID	TrainingID	ProjectID
E001	T101	P201
E001	T101	P202
E001	T102	P201
E001	T102	P202
E002	T103	P203
E002	T104	P203
E002	T104	P204
E003	T105	P205

Eliminating Multi-Valued Dependencies

- In the 'EmployeeTrainingProjects' table, the multi-valued dependencies (employees to training sessions and to projects) were jumbled together, which is not ideal in database design. This structure could lead to redundancy and make updates or queries more complex than necessary.

Functional Dependencies and Closure

Functional Dependencies

```
(EmployeeID, TrainingID, ProjectID) -> (None)
```

- The composite key (EmployeeID, TrainingID, ProjectID) uniquely identifies each record, and there are no functional dependencies where one part of this composite key determines the others.

Closure and Keys Analysis

Attribute Set	Closure	Is Superkey?	Is Key?	Anomaly Indication
EmployeeID, TrainingID, ProjectID	EmployeeID, TrainingID, ProjectID	Yes	Yes	No apparent anomaly (Unique combination of attributes)

- This shows that the combination of EmployeeID, TrainingID, and ProjectID as a whole determines each record, but none of these attributes individually or in smaller combinations determine the others.

Functional Dependencies and Closure (cont.)

Functional Dependencies

```
(EmployeeID, TrainingID, ProjectID) -> (None)
```

Closure and Keys Analysis

Attribute Set	Closure	Is Superkey?	Is Key?	Anomaly Indication	4NF Compliance
EmployeeID, TrainingID, ProjectID	EmployeeID, TrainingID, ProjectID	Yes	Yes	No apparent anomaly (Unique combination of attributes)	Not Compliant (Multi-valued dependencies are independent)

- The absence of individual functional dependencies within the composite key indicates the table is structurally compliant with BCNF. However, the presence of independent multi-valued dependencies (each EmployeeID associated with multiple TrainingIDs and ProjectIDs) challenges 4NF compliance.
- The issue in 4NF arises from the fact that training sessions and projects, although both related to an employee, do not influence each other.

Limitations

- The functional dependency and closure analysis do not directly identify or address multi-value dependencies. The check do not capture attribute independence.

Transitioning to Fourth Normal Form (4NF)

Normalisation

- Our goal is to remove multi-valued dependencies that are independent of the primary key.
- We create two separate tables: one for employee training 'EmployeeTraining' and another for employee projects 'EmployeeProjects'.

Transition into 4NF

■ EmployeeTraining

EmployeeID	TrainingID
E001	T101
E001	T102
E002	T103

■ EmployeeProjects

EmployeeID	ProjectID
E001	P201
E001	P202

Understand the Change

- By dividing the data into two focused tables, we align with 4NF by eliminating non-trivial multi-valued dependencies. Each table has a clear, singular purpose, reducing complexity and potential for data anomalies.
- This separation improves the database's scalability and adaptability. It becomes easier to add new training sessions or projects, and the data structure is more resilient to changes in one aspect of the employees' records.

Practice Problems (1 of 6)

Question: The following shows a data table in a school library tracking student book borrowings. To what stage is this table normalised?

StudentID	StudentName	Courses
1001	Alice	Math, Science
1002	Bob	History; Literature
1003	Clara	Math, Science, Literature
1004	Dave	Physics, Chemistry
1005	Ellen	Biology, History

Answer: The table is not in 1NF because the Courses column contains multiple values in a single cell. 1NF requires that each cell contain only atomic (singular) values.

Practice Problems (2 of 6)

Question: A university maintains a record of students and the courses they're enrolled in, along with the instructor's details. To what stage is this data normalized?

StudentID	CourseID	StudentName	InstructorName
1001	C101	Alice	Dr. Smith
1002	C102	Bob	Dr. Jones
1003	C103	Clara	Dr. Williams
1004	C104	Dave	Dr. Taylor

Answer: It is in 1NF as each cell contains atomic values, and there are no partial dependencies on a composite primary key.

- Is it in 2NF? 2NF requires that all non-key attributes be fully functionally dependent on the entire primary key.
- (StudentID, CourseID) is a composite primary key.
- **2NF Violation:** 'StudentName' is only dependent on 'StudentID' and not on the entire composite key ('StudentID', 'CourseID'). This means 'StudentName' can be determined solely by 'StudentID', irrespective of 'CourseID'.

Practice Problems (3 of 6)

Question: The data table keeps track of its employees and their departmental affiliations. The table also records the locations of these departments. At what stage of normalisation is this table?

EmployeeID	DepartmentID	DepartmentLocation
E001	D101	New York
E002	D102	Los Angeles
E003	D101	New York
E004	D103	Chicago

Answer: 'EmployeeID' is the primary key. The table is in 2NF because it does not have any partial dependency. Each non-key attribute ('DepartmentID', 'DepartmentLocation') is fully functionally dependent on the entire primary key ('EmployeeID').

- Is it in 3NF? In 3NF, all non-key attributes should be directly dependent on the primary key and not on another non-key attribute.
- **3NF Violation:** The table is not in 3NF due to the presence of a transitive dependency. 'DepartmentLocation' is not directly dependent on 'EmployeeID' (the primary key) but is indirectly dependent 'through DepartmentID'.

Practice Problems (4 of 6)

Question: An online store keeps track of orders and customer details in separate but related tables. What normal form are these tables in?

OrderID	ProductName	ProductPrice
0101	Widget	\$10
0102	Gizmo	\$15
0103	Doodad	\$20
0104	Thingamajig	\$25

OrderID	CustomerName	CustomerAddress
0101	Alice	123 Maple St, Denver
0102	Bob	456 Oak St, Boston
0103	Clara	789 Pine St, Seattle
0104	Dave	101 Birch St, Austin

Answer: The tables are not in 3NF. If 'ProductPrice' can be determined by knowing 'ProductName', and 'ProductName' is dependent on 'OrderID', then 'ProductPrice' is transitively dependent on 'OrderID' through 'ProductName'. Similarly for 'CustomerAddress'.

- **3NF Violation:** This transitive dependency violates the rules of 3NF. In a 3NF-compliant table, 'ProductPrice' should depend solely on the primary key ('OrderID') and not through another non-key attribute ('ProductName'). Similarly for 'CustomerAddress'.

Practice Problems (5 of 6)

Question: The data table records each doctor's specializations along with the various locations where they practice. At what stage of normalization is this table?

DoctorID	Specialization	Location
D001	Cardiology	New York
D001	Cardiology	Boston
D001	Endocrinology	New York
D002	Pediatrics	Chicago
D002	Pediatrics	Los Angeles

Answer: The table is in 3NF as each attribute is only dependent on the primary key (DoctorID), and there are no transitive dependencies. Each row is a unique combination of a doctor, their specialization, and location.

- Is it in 4NF? In 4NF, the table has no multi-valued dependencies other than trivial ones.
- **4NF Violation:** The table is not in 4NF due to the presence of multi-valued dependencies. A doctor has multiple specializations and works at multiple locations, but these two attributes are independent of each other.

Practice Problems (6 of 6)

- For a given DoctorID, there are multiple values for Specialization and Location that are unrelated to each other. This is a characteristic of non-trivial multi-valued dependencies, which violates the requirements of 4NF.
- To bring this table into 4NF, you would need to remove the multi-valued dependencies. This can be achieved by splitting the table into two:

DoctorID	Specialization
D001	Cardiology
D001	Endocrinology
D002	Pediatrics

DoctorID	Location
D001	New York
D001	Boston
D002	Chicago
D002	Los Angeles

- In these two new tables, each attribute is either fully functionally dependent on the primary key or is a primary key itself, meeting the requirements of 4NF by resolving the multi-valued dependencies.

Recap and Key Takeaways

- Normalisation directly addresses data redundancy and consistency, relying on functional dependencies and closure analysis. This systematic process reduces anomalies, ensuring data is organised logically and efficiently.
- Normal forms in database design impose stricter structural rules for higher levels, from 1NF through BCNF to 4NF. However, designers can bypass sequential progression and target higher normal forms directly, tailoring the approach to the specific needs and efficiency requirements of the database.

EmployeeID	DepartmentID	DepartmentLocation
E001	D101	New York
E002	D102	Los Angeles
E003	D101	Chicago

- An anomaly is observed in the table. The same DepartmentID, 'D101', is associated with two different locations ('New York' and 'Chicago'), indicating a violation of 3NF due to transitive dependency. This highlights the need for further normalization to remove such inconsistencies.

Preparing for Design Case Studies:

- Next Lecture Preview:
 - We transform denormalized data into efficient database schemas using ERD techniques, focusing on logical data representation and "one fact per table" design principles.
 - We'll perform a detailed analysis to identify the achieved normal form, emphasising the transition towards BCNF or 4NF.

Can you think of a scenario where normalisation might actually hinder database performance or usability?