DATA ANALYSIS

for Business Analysts

Using data to understand problems, challenge assumptions and support decisions

COURSE HANDBOOK

A practical guide to conceptual data modelling for Business Analysts

Northbridge Equipment Hire case study • Exercises • Discussions • Practical reference

How to use this handbook

This handbook is designed to stand on its own after the course. It follows the same Northbridge Equipment Hire case study, but reorganises the material into a practical reference you can return to during real analysis work.

CORE IDEA

A requirement tells us what must happen. Data analysis asks: what must the business know for that to happen?

The focus throughout is conceptual data modelling: understanding business concepts, the facts held about them, and the rules that connect them. Database tables, indexes and implementation technologies come later.

Learning outcomes

- Identify candidate business classes from requirements, processes, use cases and other evidence.
- Distinguish classes from individual objects and define meaningful attributes and identifiers.
- Discover and name associations and use multiplicity to express structural business rules.
- Recognise hidden classes in many-to-many relationships and use whole-part and generalisation structures carefully.
- Cross-check functional requirements against information using CRUD analysis and scenarios.
- Review and challenge a conceptual model with stakeholders rather than treating it as finished artwork.

Contents

1. **Understanding the business through data** (course pages 1–7)
2. **Classes, objects and attributes** (course pages 8–18)
3. **Associations, multiplicity and business rules** (course pages 19–25)
4. **Hidden classes and richer relationships** (course pages 26–31)
5. **Whole-part structures and generalisation** (course pages 32–39)
6. **Cross-checking behaviour and information** (course pages 40–47)
7. **Making the model precise** (course pages 48–55)
8. **The Northbridge final investigation** (course pages 56–59)
9. **Bringing it together** (course pages 60–62)
10. **Practical BA reference**

CHAPTER 01

Understanding the business through data

Start with requirements but look beyond behaviour to the information the business must know.

IN THIS CHAPTER

Data Analysis for Business Analysts • Building a model of the business information • Understand the information before designing its storage • Introducing Northbridge Equipment Hire • Start with a simple requirement • Nouns are clues — not automatic classes • Exercise 1 — Find the information hidden in the requirement

WELCOME

Data Analysis for Business Analysts

Every requirement says something about behaviour: something the business or system must do.
But almost every requirement also says something about **information**.

Consider a simple requirement: *"The system shall allow a member of staff to record a rental for a customer."* The action is easy to spot: **record**. But what is a **Rental**? What must we know about one? How is it related to a **Customer**? What exactly is being rented?

Those questions take us into data analysis: discovering the business concepts about which information must be held, defining their characteristics and understanding the rules that connect them.

KEY POINT

A requirement tells us what must happen. Data analysis asks: what must the business know for that to happen?

WHAT YOU WILL LEARN

Building a model of the business information

During this course you will progressively analyse the requirements for a fictional equipment-rental business. You will not be given a finished data model. You will **discover it as the requirements emerge**.

By the end of the course you should be able to:

- identify candidate business classes from requirements, processes, use cases and other evidence;
- distinguish classes from individual objects;
- define meaningful attributes and identifiers;
- discover and name associations between classes;
- use multiplicity to express business rules;

- recognise optionality, constraints and integrity rules;
- investigate many-to-many associations and uncover hidden classes;
- use whole-part relationships and generalisation where they clarify the business;
- cross-check functional requirements against the class model using CRUD analysis; and
- develop a conceptual model that can later inform logical and physical data design.

WHY DATA ANALYSIS?

Understand the information before designing its storage

During analysis, our concern is the **logical business information**: the things the business needs to know about, the facts it needs to record about them, and the rules governing their relationships.

We are not yet deciding how many database tables will exist, what indexes they will use, which programming language will implement the system, or how a particular package stores its data.

A good conceptual model gives stakeholders and developers a shared vocabulary, exposes ambiguity, reduces unnecessary duplication and makes important business rules visible. It may eventually provide a strong foundation for a database schema — but that is a later design step.

KEY POINT

Model the business meaning first.
Decide the physical storage later.

CASE STUDY

Introducing Northbridge Equipment Hire

Northbridge Equipment Hire rents tools and small construction equipment to customers. Staff register customers, reserve equipment, record rentals, issue and receive equipment, record damage and take payments.

Northbridge currently relies on a mixture of spreadsheets, paper forms and staff knowledge. A new rental-management system is being investigated.

You are the Business Analyst. As requirements emerge, you will build a conceptual class model of the information the business needs.

KEY POINT

The model you finish with should be richer than the model you start with.
That is not rework. It is analysis .

REQUIREMENT R1

Start with a simple requirement



Figure 5: Northbridge model at this stage

REQUIREMENT / PROMPT

R1 — The system shall allow a member of staff to record a rental for a customer.

Functional analysis naturally notices the verb **record**. Data analysis notices the business concepts around that action.

Rental is clearly important. So is **Customer**. They are candidate classes: types of things about which Northbridge appears to need persistent information.

We should not yet invent every possible field or relationship. Our first model can be deliberately small.

BA INSIGHT

Keep the first model deliberately small. The purpose is to expose questions, not to guess the final answer.

FINDING CLASSES

Nouns are clues — not automatic classes

Requirements, process descriptions, use cases, forms and stakeholder conversations contain nouns and noun phrases. These are useful places to look for candidate classes.

But blindly turning every noun into a class produces poor models. A requirement might mention a screen, a button, a list, a system, a date and a member of staff. Some may matter to the business information model; others may not.

A candidate class should survive questions such as:

- Does the business recognise this concept?
- Do we need to remember information about it?
- Can we describe it clearly?
- Does it have meaningful characteristics?

- Can there be more than one identifiable occurrence?
- Does it have meaningful relationships with other business concepts?

KEY POINT

Find candidates freely. Validate them carefully.

YOUR TURN

Exercise 1 — Find the information hidden in the requirement

REQUIREMENT / PROMPT

R1 — The system shall allow a member of staff to record a rental for a customer.

Identify the candidate business concepts in this requirement. Then decide which you would put into the class model now.

For each candidate, ask:

- Is it a business concept or merely wording about the solution?
- Would Northbridge need to retain information about individual occurrences?
- Can you describe what it means without referring to the software?

Which two classes are strongly justified by the evidence we currently have?

CHAPTER 02

Classes, objects and attributes

Turn evidence into candidate concepts, then decide what each concept means and what the business must know about it.

IN THIS CHAPTER

Customer and Rental are our strongest candidates • A class describes a type of thing • Name classes in the language of the business • What must we know about each occurrence? • Attributes come from investigation — not imagination • How do we distinguish one occurrence from another? • An attribute needs meaning, not merely a name • Must this value be stored, or can it be derived? • Processes tell us what happens to business information • Exercise 2 — Mine the process model for information • The process expands our candidate model

EXERCISE 1 — DISCUSSION

Customer and Rental are our strongest candidates

Customer is a business concept Northbridge recognises independently of the proposed system. Northbridge needs to distinguish one customer from another and retain information about them.

Rental is also a recognisable business concept. There will be many rentals, each of which must be distinguished from the others and associated with relevant business information.

System and **record** do not become classes merely because they appear in the sentence. “Record” describes behaviour. “System” describes the solution boundary.

```
Customer -------- Rental
```

KEY POINT

Our model is incomplete — and that is perfectly acceptable.
We add detail when investigation gives us evidence for it.

CLASSES AND OBJECTS

A class describes a type of thing

A **class** describes a set of business objects that share the same kinds of characteristics and relationships. An **object** is one particular instance.

```
CLASS: Customer
OBJECTS:
Customer 1042 – Greenfield Builders Ltd
Customer 1043 – A. Rahman
Customer 1044 – Holt Engineering
```

The class tells us what information every Customer may have. Each Customer object holds its own values for that information.

Business Analysts normally build the enduring structure at class level. Specific objects are useful when we want to test whether the model makes sense in a real scenario.

MODELLING LANGUAGE

Name classes in the language of the business

A conceptual model is a communication tool. Class names should therefore use clear, consistent business terminology — normally singular noun phrases such as **Customer**, **Rental** and **Equipment Item**.

Avoid premature technical names such as *tblCustMaster*, *RentalRecord* or *CustomerDTO*. Those belong to possible implementations, not to the business concept.

Naming disagreements can reveal genuine requirements problems. If one department says *Customer*, another says *Account*, and a third uses both terms differently, do not standardise the vocabulary until you understand the distinction.

KEY POINT

A shared data model is also a shared business vocabulary.

ATTRIBUTES

What must we know about each occurrence?



Figure 11: Northbridge model at this stage

Once a class has been identified, we investigate its **attributes**: the characteristics that describe an individual object of that class.

For Customer, investigation might reveal: Attributes should be coherent with the class. A customer's name belongs naturally to Customer. A rental start date does not: it describes a **particular** Rental, not the Customer.

KEY POINT

Ask: “Is this a fact about the class itself, or about something related to it?”

DISCOVERING ATTRIBUTES

Attributes come from investigation — not imagination

Useful sources include requirements, business forms, existing screens, reports, interfaces, business rules, data dictionaries, regulations and stakeholder interviews.

Existing artefacts are evidence, not unquestionable truth. A paper form may contain fields that are obsolete, duplicated or included only because an old system required them. Conversely, staff may know important information that has never been formally recorded.

For each proposed attribute ask why the business needs it, what it means, who supplies it, when it becomes known, whether it can change, and whether it is mandatory.

IDENTITY

How do we distinguish one occurrence from another?

If Northbridge has many Customers and many Rentals, the business must be able to distinguish one occurrence from another. This leads us to consider **identifiers**.

A customer name is unlikely to be sufficient: two customers may have the same name, and a customer's name may change. Northbridge may therefore assign a stable Customer Number.

At conceptual level we are interested in the business requirement for identity. We are not yet choosing database primary-key technologies.

KEY POINT

“How do you know this is the same Customer?” is a business question before it is a database question.

ATTRIBUTE RULES

An attribute needs meaning, not merely a name

A useful attribute definition may include its meaning, allowable values, whether it is mandatory, and relevant validation rules.

Simple analysis-level types such as **text**, **number**, **date**, **currency** and **boolean** may be sufficient. We do not

need implementation-specific types yet.

```
rentalStartDate : date
agreedDailyRate : currency
depositRequired : Boolean
```

A model becomes much more useful when accompanied by definitions. “Status” is a poor attribute specification until everybody agrees what statuses exist and what each one means.

DERIVED INFORMATION

Must this value be stored, or can it be derived?

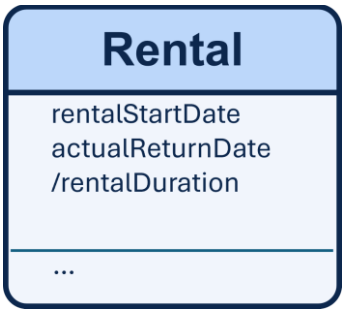


Figure 15: Northbridge model at this stage

Some information can be calculated from other information. A customer's age can be derived from date of birth and today's date. A rental duration may be derived from start and return dates.

UML can mark a derived attribute with a slash. Identifying derivation prevents duplicated facts from drifting out of agreement and exposes the business rule behind the calculation.

PROCESS MODELS AS EVIDENCE

Processes tell us what happens to business information

Suppose the process model contains these activities:

```
Register Customer
Check Equipment Availability
Record Rental
Issue Equipment
Accept Return
Record Damage
Take Payment
```

The verbs describe processing. The nouns point towards information that the process consumes, creates or changes.

Record Rental strengthens our Rental class.
Issue Equipment introduces Equipment as a candidate.

Record Damage may introduce Damage — but we still need to investigate whether Damage is a class, an attribute, an event or something else.

YOUR TURN

Exercise 2 — Mine the process model for information

```
Register Customer
Check Equipment Availability
Record Rental
Issue Equipment
Accept Return
Record Damage
Take Payment
```

Identify candidate classes suggested by these activities. Do not automatically promote every noun to a class.

For each candidate, write one sentence describing what it means to Northbridge. If you cannot describe it clearly, mark it for investigation.

Which new business concepts would you add to the model with reasonable confidence?

EXERCISE 2 — DISCUSSION

The process expands our candidate model

Strong candidates now include:

- Customer — a person or organisation hiring equipment;
- Rental — an agreement under which equipment is hired;
- Equipment Item — a particular physical item available for hire;
- Payment — a financial transaction received against an amount owed;
- Damage Report — potentially a record of damage discovered or reported.

“Availability” may be derived from reservations, rentals, maintenance and status rather than being a separate business class. “Return” may initially look like an event affecting Rental and Equipment Item rather than a persistent class of its own.

KEY POINT

Candidate classes may appear, disappear, split or merge as understanding improves.

CHAPTER 03

Associations, multiplicity and business rules

A line between boxes is not the point. The meaning of the relationship - and how many can participate - is the business rule.

IN THIS CHAPTER

Classes become useful when we understand how they relate • Read every association in both directions • How many? Multiplicity captures a business rule • Multiplicity makes our first relationship precise • Exercise 3 — Turn statements into multiplicities • Multiplicity is compact requirements documentation • A many-to-many association may be perfectly valid

ASSOCIATIONS

Classes become useful when we understand how they relate



Figure 19: Northbridge model at this stage

An **association** represents a meaningful business relationship between classes. It is not enough to draw lines because two concepts appear in the same requirement.

Our first requirement implies a relationship between Customer and Rental.

Naming the association as a verb phrase allows us to verbalise and validate it with stakeholders: *"A Customer makes a Rental."*

KEY POINT

If the business cannot explain what a line means, the line probably should not be there.

BA INSIGHT

Name the relationship so that it can be read aloud as a business statement. If the wording is awkward, the analysis may still be unclear.

VALIDATING ASSOCIATIONS

Read every association in both directions

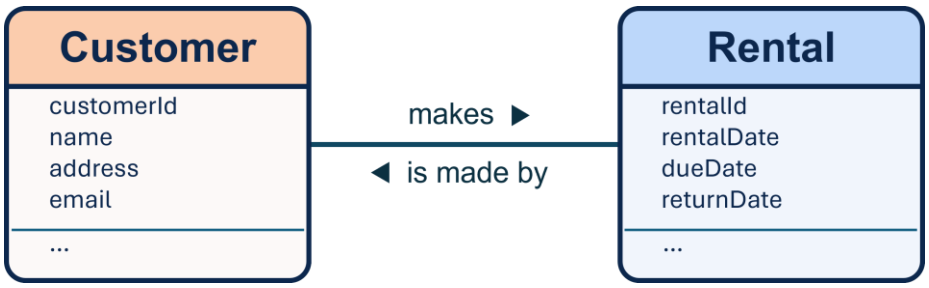


Figure 20: Northbridge model at this stage

Suppose we model:

We should be able to verbalise both directions:

- Each Customer makes Rental(s).
- Each Rental is made by a Customer.

The wording often exposes missing rules. Can a Rental exist before the Customer is identified? Can a corporate customer have many concurrent rentals? Can a Rental be transferred between customers?

KEY POINT

The diagram is not decoration. It is a device for asking better questions.

MULTIPLICITY

How many? Multiplicity captures a business rule

Multiplicity states how many objects of one class may be associated with one object of another class. Common values include:

```
1       exactly one
0..1    zero or one
0..*    zero or many
1..*    one or many
```

The notation is compact, but the meaning is business-critical. It can determine whether the eventual system permits or prevents particular situations.

MULTIPLICITY

Multiplicity makes our first relationship precise

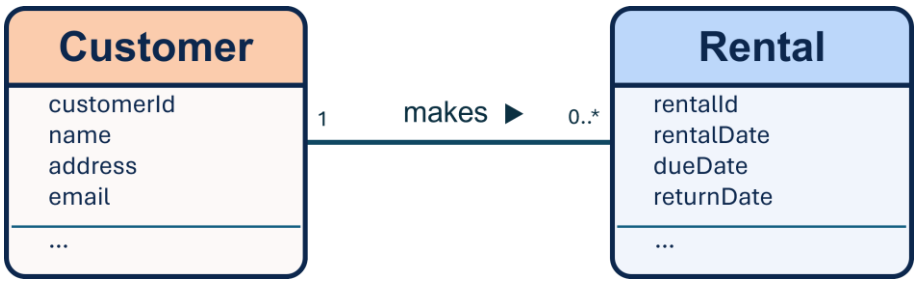


Figure 22: Northbridge model at this stage

After talking to Northbridge staff, we learn:

- a Customer may be registered before making any Rental;
- a Customer may make many Rentals over time; and
- every Rental must belong to exactly one Customer.

Read from Rental towards Customer: *Each Rental is made by one and only one Customer*.

Read from Customer towards Rental: *Each Customer may make zero or many Rentals*.

BA INSIGHT

Multiplicity is a claim about the business. Challenge it with examples and exceptions rather than treating it as notation.

YOUR TURN

Exercise 3 — Turn statements into multiplicities

Northbridge tells you:

1. Every Equipment Item is of exactly one Equipment Type.
2. An Equipment Type may exist before Northbridge owns any physical items of that type.
3. A Rental must contain at least one Equipment Item.
4. An Equipment Item may be rented many times over its lifetime.

Sketch the associations and add multiplicities. Read each relationship in both directions before moving on.

EXERCISE 3 — DISCUSSION

Multiplicity is compact requirements documentation

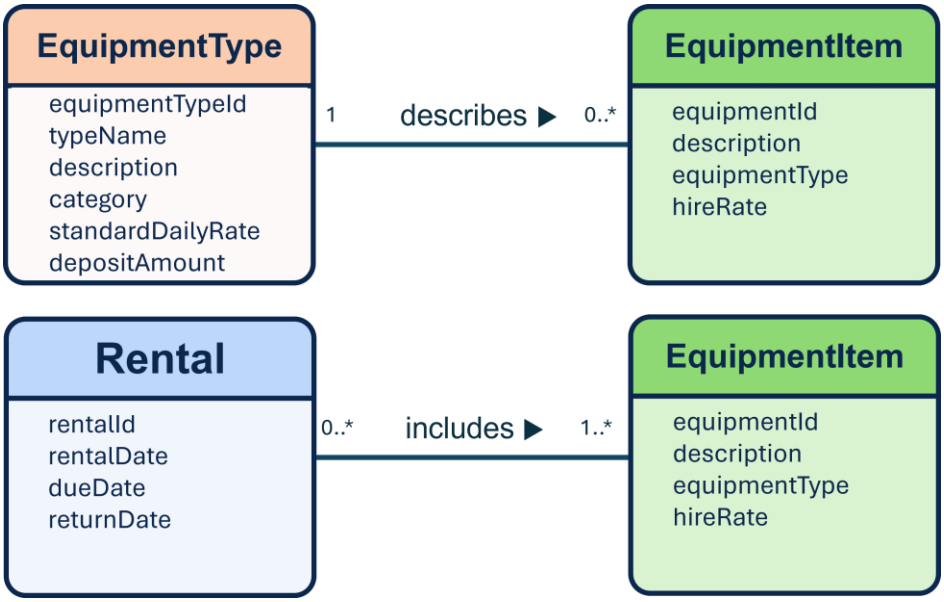


Figure 24: Northbridge model at this stage

The first relationship is straightforward: every physical item has exactly one type, while a type can exist with no physical stock yet.

The Rental-to-EquipmentItem relationship is more interesting. A Rental contains one or more items, while an Equipment Item may participate in many Rentals over time. That gives us a **many-to-many association**.

KEY POINT

Do not “fix” the many-to-many yet. First ask what it means to the business.

MANY-TO-MANY ASSOCIATIONS

A many-to-many association may be perfectly valid

In a conceptual class model, many-to-many associations are legitimate. An Equipment Item may appear on many Rentals over time, and a Rental may contain several Equipment Items.

But a many-to-many relationship deserves investigation because the **relationship itself** may have information that the business needs to record.

For *this particular Equipment Item on this particular Rental*, is there anything Northbridge needs to know?

CHAPTER 04

Hidden classes and richer relationships

Difficult relationships often tell us that the business contains another concept we have not yet named.

IN THIS CHAPTER

The relationship has information of its own • Exercise 4 — Look for the class hiding in the relationship • Accreditation belongs to the relationship • The same two classes may be related in more than one way • Sometimes a class relates to itself • Not every business rule fits into multiplicity

DISCOVERING A HIDDEN CLASS

The relationship has information of its own

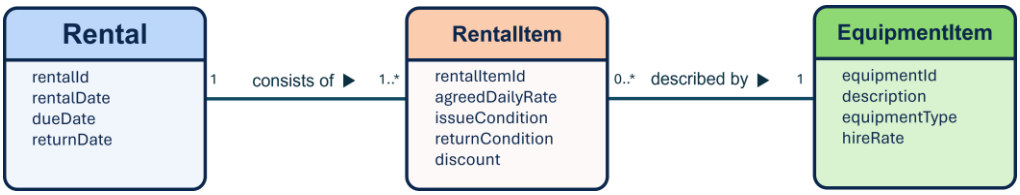


Figure 26: Northbridge model at this stage

Northbridge says that for each item included in a rental it records:

- the agreed daily rate;
- the condition when issued;
- the condition when returned; and
- any item-specific discount.

Those facts belong neither to Equipment Item in general nor to Rental in general. They describe **one item's participation in one rental**.

We have discovered another class: **Rental Item**.

BA INSIGHT

Do not create an association class merely because a database might need a linking table. Add it when the business has information about the relationship itself.

YOUR TURN

Exercise 4 — Look for the class hiding in the relationship

Northbridge staff members can operate many types of specialist equipment. Each Equipment Type can be operated by many Staff Members.

For each Staff Member / Equipment Type combination, Northbridge records:

- accreditation level;
- date achieved;
- expiry date; and
- who authorised the accreditation.

What class is hiding in the many-to-many relationship? What would you call it?

EXERCISE 4 — DISCUSSION

Accreditation belongs to the relationship



Figure 28: Northbridge model at this stage

Accreditation is a strong candidate class because its attributes describe a particular Staff Member's competence for a particular Equipment Type.

This is a useful general pattern. When a relationship has facts of its own, those facts may reveal a business concept that was previously hidden.

KEY POINT

Do not resolve many-to-many associations merely because a database eventually will. Resolve them during analysis when the business meaning reveals another class.

ASSOCIATION ROLES

The same two classes may be related in more than one way



Figure 29: Northbridge model at this stage

Two classes can participate in several distinct business relationships. For example, a Staff Member might **create** a Damage Report, while another Staff Member later **authorises** it.

Drawing only one vague line between Staff Member and Damage Report would lose important meaning. Each association should represent a recognisable business relationship. Association names and role names help us

distinguish those meanings.

BA INSIGHT

Two classes can participate in more than one distinct relationship. Name the roles so that different business meanings remain visible.

REFLEXIVE ASSOCIATIONS

Sometimes a class relates to itself

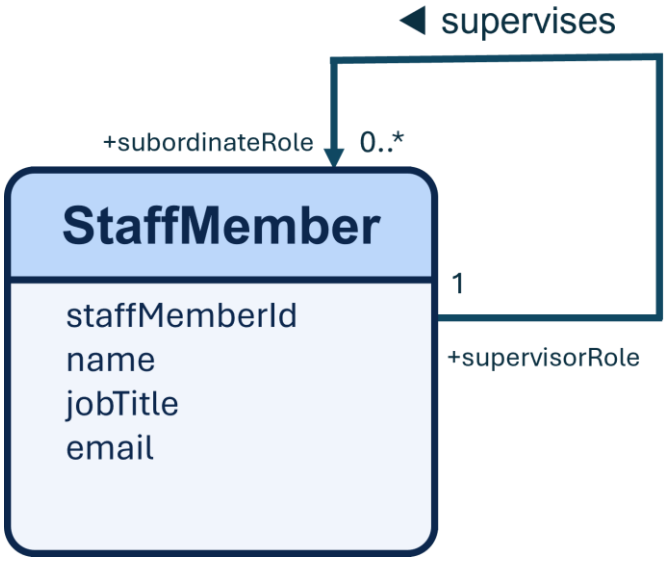


Figure 30: Northbridge model at this stage

Northbridge tells us that one Staff Member may supervise other Staff Members.

We do not need a separate class called Supervisor merely because somebody performs that role. Instead, Staff Member can have a relationship with itself.

KEY POINT

Roles describe how objects participate in a relationship.
A role is not automatically a new class.

CONSTRAINTS

Not every business rule fits into multiplicity

Multiplicity captures many structural rules, but businesses also impose constraints that need additional expression.

Northbridge might state:

- a Customer may have either a standard credit arrangement or a prepaid arrangement, but not both;
- items on a Rental must be returned before that Rental can be closed;
- only appropriately accredited Staff Members may issue certain Equipment Types.

Some constraints can be shown directly on the model; others belong in supporting business-rule documentation. The important point is that the model helps us discover the rule.

CHAPTER 05

Whole-part structures and generalisation

Use structure only when it clarifies genuine business meaning: ownership, lifetime, and 'is a kind of'.

IN THIS CHAPTER

Some associations describe containment • Exercise 5 — Whole-part or ordinary association? • Use whole-part notation only when the business meaning justifies it • When several classes are kinds of the same thing • Use the “is a” test • Some classes exist to capture common meaning • Exercise 6 — Association or generalisation? • Generalisation and association answer different questions

WHOLE AND PART

Some associations describe containment



Figure 32: Northbridge model at this stage

A whole-part relationship deserves special attention. In UML, **aggregation** can describe a looser whole-part relationship, while **composition** expresses strong ownership and lifetime dependency.

Rental and Rental Item are a useful example. A Rental Item exists only as part of a particular Rental. It has no independent business existence away from that Rental.

By contrast, an Equipment Item exists independently of any particular Rental and can participate in many Rentals during its life.

YOUR TURN

Exercise 5 — Whole-part or ordinary association?

Consider these Northbridge relationships:

1. Rental and Rental Item
2. Branch and Equipment Item
3. Customer and Rental
4. Equipment Type and Equipment Item

Which, if any, would you model as strong composition? Which objects have meaningful independent lives?

Avoid choosing notation because it “looks right”. What business rule about ownership or lifetime are you expressing?

EXERCISE 5 — DISCUSSION

Use whole-part notation only when the business meaning justifies it

Rental / Rental Item is the strongest composition candidate: a Rental Item represents part of one particular Rental and has no useful independent existence.

Equipment Items can move between branches, so Branch / Equipment Item is better treated as an ordinary association unless Northbridge's rules say otherwise.

Customer and Rental are related but a Rental is not literally a constituent part of a Customer. Equipment Type describes Equipment Items; again, that is not necessarily a lifetime-controlled whole-part relationship.

KEY POINT

UML notation should document a rule you understand — not create a rule you have never investigated.

GENERALISATION

When several classes are kinds of the same thing

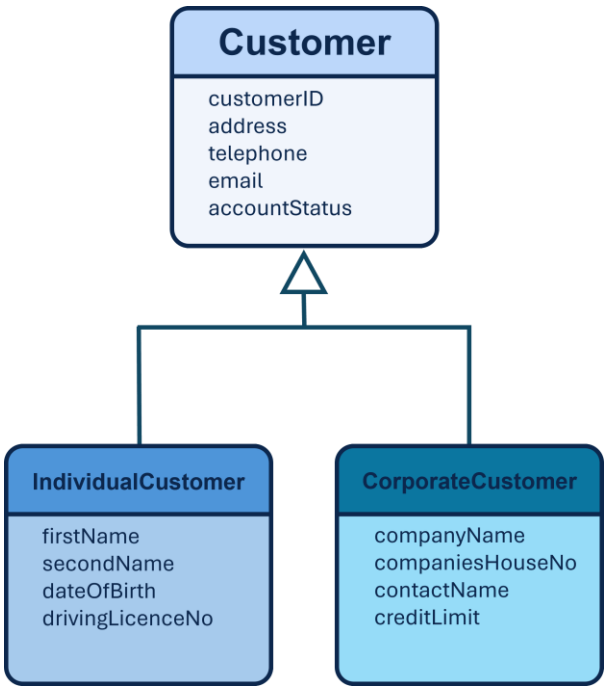


Figure 35: Northbridge model at this stage

Northbridge rents to both private individuals and organisations. Investigation reveals that both have common information such as Customer Number, contact details and account status, but each also has specialist information.

We might model **Individual Customer** and **Corporate Customer** as specialised kinds of **Customer**.

The specialised classes inherit the common characteristics and relationships of Customer and add only what is specific to them.

BA INSIGHT

Generalisation should express 'is a kind of'. Common information belongs in the general class; specialised information stays with the subtype.

GENERALISATION

Use the “is a” test

Generalisation should represent genuine specialisation:

an Individual Customer **is a** Customer;
a Corporate Customer **is a** Customer.

A Rental is **not** a Customer. An Equipment Item is **not** an Equipment Type. **Those** concepts require associations, not inheritance.

Do not use generalisation simply because two classes happen to share several attributes. They must represent specialised forms of the same business concept.

KEY POINT

Shared fields are evidence to investigate.
Shared business meaning is what justifies generalisation.

ABSTRACTION

Some classes exist to capture common meaning

A superclass can be **abstract**: useful in the model but never instantiated directly.

Suppose Northbridge's rule is that every Customer must be either an Individual Customer or a Corporate Customer. There is never an occurrence that is merely “a Customer” with no subtype. Customer may then be treated as abstract.

Abstract classes allow us to place common attributes, associations and rules in one place while keeping specialised information in the subclasses.

YOUR TURN

Exercise 6 — Association or generalisation?

Decide how you would model each statement:

1. A Corporate Customer is a Customer.
2. An Equipment Item is of an Equipment Type.
3. A Staff Member works at a Branch.
4. An Electric Tool is an Equipment Type.
5. A Rental is made by a Customer.

For each one, ask whether it expresses **"is a kind of"** or a relationship between different concepts.

EXERCISE 6 — DISCUSSION

Generalisation and association answer different questions

- Corporate Customer → Customer: generalisation.
- Equipment Item → Equipment Type: association.
- Staff Member → Branch: association.
- Electric Tool → Equipment Type: potentially generalisation, if Electric Tool is genuinely a specialised type in Northbridge's business taxonomy.
- Rental → Customer: association.

KEY POINT

The model should express the business semantics, not merely connect boxes.

CHAPTER 06

Cross-checking behaviour and information

Use cases, CRUD and concrete object snapshots let us test whether the information model really supports the work.

IN THIS CHAPTER

Use cases tell us which information behaviour needs • Create, Read, Update, Delete • Cross-check behaviour against information • Let the models challenge each other • Exercise 7 — Diagnose the CRUD gap • Something is missing or misunderstood • Use objects to test whether the class model works • Track business objects through a process

USE CASES AS EVIDENCE

Use cases tell us which information behaviour needs

Northbridge identifies use cases including:

```
Register Customer
Amend Customer Details
Create Rental
Add Item to Rental
Issue Rental
Record Return
Record Damage
Take Payment
Close Rental
```

Each use case should interact with business information represented in the class model. As we walk through a use case, we can ask which objects are created, read, updated or deleted.

This gives us a powerful way of cross-checking the functional and data views of the requirements.

CRUD ANALYSIS

Create, Read, Update, Delete

During its life, business information is created, read, updated and sometimes deleted. These four actions are commonly abbreviated to **CRUD**.

```
Register Customer       → CREATE Customer
Amend Customer Details  → READ + UPDATE Customer
Create Rental   → READ Customer + CREATE Rental
Close Rental    → READ + UPDATE Rental
```

CRUD analysis is not merely documentation. It allows two different models of the requirements to test one another for completeness.

CRUD MATRIX

Cross-check behaviour against information

A CRUD matrix places use cases on one axis and classes on the other. Each cell records how the use case interacts with that class.

Use CaseCustomerRentalRental ItemPaymentRegister CustomerCCreate RentalRCAdd Item to RentalRCTake PaymentRRC

Empty patterns and suspiciously similar patterns often reveal missing requirements, missing classes or overlapping use cases.

CRUD AS A DIAGNOSTIC

Let the models challenge each other

If the class model contains **Damage Report** but no use case ever creates one, either a functional requirement is missing, or the class does not belong within scope.

If a use case called **Record Return** does not read or update any relevant class, perhaps the data model is missing something — or the use case has not been understood.

If two use cases have nearly identical CRUD patterns, they may overlap. If one use case creates, updates and deletes many unrelated classes, it may be too broad.

KEY POINT

Requirements models should not live in isolation.
The strongest analysis comes when one model is used to test another.

YOUR TURN

Exercise 7 — Diagnose the CRUD gap

Your class model contains:

```
Customer
Rental
RentalItem
EquipmentItem
DamageReport
Payment
```

The current use cases Create, Read and Update every class except **DamageReport**, which is only ever Read.

What questions should this make you ask?

EXERCISE 7 — DISCUSSION

Something is missing or misunderstood

If Damage Report genuinely belongs within the system's scope, something must cause a Damage Report to come into existence. We may have omitted a use case such as **Record Damage**.

Alternatively, perhaps Damage Report is imported from another system and therefore is not created by functionality inside our scope. That boundary should be explicit.

A third possibility is that Damage Report should not be a class at all. CRUD does not tell us which answer is correct; it tells us where to investigate.

KEY POINT

A modelling inconsistency is a question generator.

TESTING THE MODEL

Use objects to test whether the class model works

A class diagram describes the possible structure over time. An **object diagram** shows particular instances and their links at one moment.

```
greenfield:Customer
rental5821:Rental
drill117:EquipmentItem
rentalItem1:RentalItem
```

By constructing a real scenario, we can test whether the model can represent it. This is especially useful for complicated multiplicities, optional relationships and unusual cases.

KEY POINT

Class diagrams describe the rules.
Object diagrams test those rules with examples.

PROCESS AND DATA TOGETHER

Track business objects through a process

Activity models can show important objects flowing through business activities and changing state.

A Rental might progress conceptually through states such as:

```
Rental [Draft]
↓ Issue Rental
Rental [Active]
↓ Accept Return
Rental [Returned]
↓ Complete Charges
Rental [Closed]
```

This connects static information structure with dynamic behaviour. It can reveal missing attributes, states, rules or requirements.

CHAPTER 07

Making the model precise

Definitions, time, controlled vocabulary, scope and stakeholder review turn a plausible diagram into a useful analysis model.

IN THIS CHAPTER

A class model is full of requirements • The diagram is not enough • Watch for different names — and identical names • Ask where a fact really belongs • Do we need the current value — or its history? • Is this merely a value — or a managed business concept? • The business world is bigger than your model • Walk the model with the people who know the business

BUSINESS RULES

A class model is full of requirements

A well-built class diagram is not simply a picture of nouns. Its structure records business rules.

- Every Rental belongs to exactly one Customer.
- A Customer may exist without ever making a Rental.
- Every Rental contains at least one Rental Item.
- Every Rental Item refers to exactly one Equipment Item.
- A Rental Item cannot exist independently of its Rental.
- An Equipment Item is of exactly one Equipment Type.

These rules need stakeholder validation just as textual requirements do.

KEY POINT

Multiplicity, optionality and structure are requirements.

BA INSIGHT

Every multiplicity and relationship is a business claim. Stakeholder disagreement is useful evidence that the model needs investigation.

DEFINING THE MODEL

The diagram is not enough

A diagram shows structure efficiently, but each class, attribute and important association should have a clear definition.

A lightweight data dictionary might record:

- business name;

- definition;
- aliases or prohibited synonyms;
- important attributes and their meanings;
- identifier;
- source or owner;
- business rules and constraints; and
- relevant requirement references.

This is particularly valuable when the same word means different things to different stakeholders.

BUSINESS VOCABULARY

Watch for different names — and identical names

Two departments may use different names for the same concept: *Client* and *Customer* may be synonyms.

More dangerously, two groups may use the same word for different concepts. “Item” might mean an Equipment Type to one person and a specific physical Equipment Item to another.

The model forces those ambiguities into the open. Resolve terminology with stakeholders rather than silently choosing whichever word appears most often.

KEY POINT

Vocabulary disagreements are often requirements disagreements in disguise.

STRUCTURING INFORMATION

Ask where a fact really belongs

Business Analysts do not need to turn conceptual class modelling into a database-normalisation exercise, but they should recognise duplicated facts and misplaced attributes.

If Customer Name is repeated on every Rental, ask whether the name is really a fact about each Rental or a fact about Customer. If Equipment Type description is repeated on every physical Equipment Item, ask the same question.

Separating stable business concepts and relating them explicitly reduces contradictory copies of the same fact and makes integrity rules clearer.

KEY POINT

Ask where a fact belongs before asking where a field will be stored.

HISTORY AND TIME

Do we need the current value — or its history?

Requirements often hide a temporal question. Suppose a Customer changes address. Should old Rentals display the new address, or the address to which equipment was delivered at the time?

Suppose an Equipment Type's standard daily price changes. Should an existing Rental be recalculated, or must it retain the agreed price?

These questions reveal whether we need historical facts, effective dates or transaction-specific values. They can change the class model significantly.

KEY POINT

"What is the value?" and "What was the value when this happened?" are different requirements.

REFERENCE DATA

Is this merely a value — or a managed business concept?

A field such as **damageType** might initially look like simple text. Investigation may reveal that Northbridge maintains an approved set of Damage Types, each with a code, description, severity and inspection rule.

At that point Damage Type may deserve to become a class in its own right. The same can happen with Equipment Type, Payment Method, Branch, Customer Category and other controlled business vocabularies.

KEY POINT

An attribute can become a class when the business needs to manage information about the value itself.

SCOPE

The business world is bigger than your model

Northbridge interacts with banks, manufacturers, insurers and external payment providers. That does not mean the rental system needs a detailed model of every concept those organisations manage.

Include enough information to support the requirements and business rules within scope. External concepts may appear as classes if Northbridge needs to retain meaningful information about them, but modelling the entire surrounding world creates noise rather than understanding.

KEY POINT

A conceptual model should be complete for its purpose — not complete for the universe.

REVIEWING THE MODEL

Walk the model with the people who know the business

A class diagram should be reviewable in ordinary business language. Point to each class and ask what it means. Read each association aloud. Challenge every multiplicity with examples and exceptions.

Useful review questions include:

- Can this occur before that exists?
- Can there ever be more than one?
- What happens in this exceptional case?
- Where would we record this fact?
- How do we distinguish two occurrences?
- What creates, changes and ends this information?

KEY POINT

The purpose of the model is not to survive review.
The purpose is to make review productive.

BA INSIGHT

Review the model by turning notation back into plain business statements. The objective is productive disagreement, not a diagram that survives untouched.

CHAPTER 08

The Northbridge final investigation

Bring the techniques together by extending the model for reservations, conversion to rental and cancellation rules.

IN THIS CHAPTER

Extend the Northbridge model • Reservation becomes a significant business concept • Reservation Item may emerge from the relationship • The new requirements also reveal rules and lifecycle

FINAL INVESTIGATION

Extend the Northbridge model

Northbridge gives you three new requirements:

REQUIREMENT / PROMPT

R12 — A customer may reserve one or more equipment items for a future period. R13 — A reservation may be converted into a rental when the equipment is collected. R14 — If a customer cancels a reservation after the cancellation deadline,
a cancellation charge may be applied.

Extend the conceptual model. Consider:

- new candidate classes;
- attributes;
- associations and multiplicities;
- whether a many-to-many relationship hides another class;
- business rules and temporal information;
- how Reservation relates to Rental; and
- which use cases would CRUD the new information.

FINAL INVESTIGATION — DISCUSSION

Reservation becomes a significant business concept

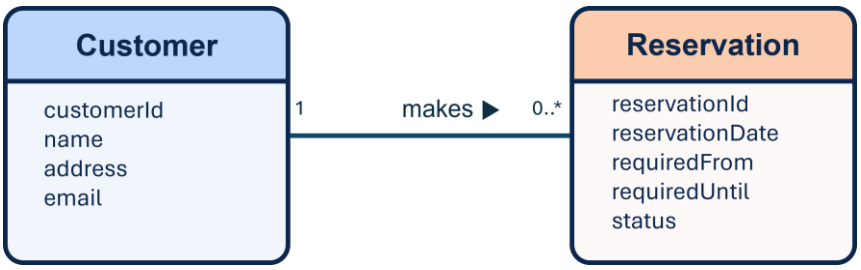


Figure 57: Northbridge model at this stage

Reservation is clearly a candidate class. It has its own identity and lifecycle, exists before any Rental is created, and has information such as reservation date, required-from date, required-to date and status.

A Customer may make many Reservations, while each Reservation belongs to one Customer.

A Reservation can cover several Equipment Items, while an Equipment Item can be reserved many times for different periods. That gives us another many-to-many relationship to investigate.

FINAL INVESTIGATION — DISCUSSION

Reservation Item may emerge from the relationship



Figure 58: Northbridge model at this stage

If Northbridge needs information about a particular item within a particular reservation — for example agreed rate, reserved period, priority or fulfilment status — the relationship has information of its own.

We should not invent Reservation Item merely because a relational database may eventually need a linking table. We add it because the **business analysis** identifies information about that relationship.

FINAL INVESTIGATION — DISCUSSION

The new requirements also reveal rules and lifecycle

Reservation may be converted into a Rental, but we need to investigate what “converted” means. Does the Reservation remain for audit history? Can one Reservation generate more than one Rental? Can only part of a Reservation be fulfilled?

The cancellation rule introduces further information: a cancellation deadline, cancellation time, charge and perhaps a Charge concept. The amount may depend on when the cancellation occurs.

These are not implementation details. They are questions about the meaning and lifecycle of business information.

KEY POINT

Every new requirement is an opportunity to challenge and enrich the information model.

BA INSIGHT

Try to break the model. Ask what happens in exceptions, partial fulfilment and awkward lifecycle states.

CHAPTER 09

Bringing it together

The model is not a monument. It is a disciplined way to ask better questions and discover requirements.

IN THIS CHAPTER

A practical approach to data analysis • The class diagram is not the end product of requirements analysis • Data Analysis for Business Analysts

BRINGING IT TOGETHER

A practical approach to data analysis

1. Investigate requirements and business activity.
2. Identify candidate business concepts. Treat nouns as clues, not answers.
3. Validate classes. Define what they mean to the business.
4. Discover attributes and identity. Ask what must be known about each occurrence.
5. Discover associations. Name and verbalise meaningful relationships.
6. Define multiplicities and constraints. Capture structural business rules.
7. Investigate difficult relationships. Look for hidden classes and roles.
8. Use generalisation and whole-part structures carefully.
9. Cross-check processing and information. Use CRUD and scenarios.
10. Iterate. Let new requirements change the model.

ONE LAST THOUGHT

The class diagram is not the end product of requirements analysis

It is tempting to think that analysts gather the requirements first and draw the data model afterwards. In practice, building the model is one of the ways we **discover the requirements**.

- Asking what a Rental is may reveal missing business rules.
- Asking what it relates to may reveal missing requirements.
- Asking how many may reveal exceptions.
- Asking where an attribute belongs may reveal that two apparently simple concepts are actually different.

KEY POINT

The class diagram is not something you draw after requirements analysis. Building it is part of requirements analysis.

COURSE COMPLETE

Data Analysis for Business Analysts

You have taken a small functional requirement and progressively developed a conceptual model of the business

information needed to support a much richer set of requirements.

The next time you encounter a requirement such as *Record Rental*, *Create Booking*, *Approve Claim* or *Register Applicant*, look beyond the verb.

KEY POINT

Ask what the noun means. Ask what must be known about it. Ask how it relates to everything else. That is data analysis.

PRACTICAL REFERENCE

A Business Analyst's data-modelling checklist

Use these questions during workshops, requirements reviews or whenever a process description seems deceptively simple.

Meaning: What are the important business concepts? What does each one mean to the business?

Evidence: Which requirements, forms, reports, screens, interviews or process steps suggest that the concept matters?

Attributes: What must the business know about each occurrence? Does each attribute describe this concept, or something else?

Identity: How does the business distinguish one occurrence from another?

Relationships: Which concepts are related, and what does the relationship mean in plain English?

Multiplicity: How many can participate at each end? Can there be none? Exactly one? Many?

Hidden concepts: Does a many-to-many relationship carry facts of its own? If so, is another business concept hiding in the relationship?

Whole-part: Does the part have an independent business life, or does it belong to the whole?

Generalisation: Is one concept genuinely a kind of another? What information is common, and what is specialised?

Time: Is this fact true now, true at the time of an event, or something whose history must be retained?

Vocabulary: Are synonyms or homonyms hiding disagreement? Are statuses and categories controlled business vocabulary?

Behaviour: Which use cases create, read, update or retire the information?

Scope: Are we modelling enough to understand the problem without trying to model the entire organisation?

Review: Can stakeholders read the model as business statements? Where do they disagree?

Multiplicity quick reference

Notation	Business meaning
1	exactly one
0..1	zero or one
0..*	zero or many
1..*	one or many

Ten-step practical approach

1. Investigate requirements and business activity.
2. Identify candidate business concepts. Treat nouns as clues, not answers.
3. Validate classes. Define what they mean to the business.
4. Discover attributes and identity. Ask what must be known about each occurrence.
5. Discover associations. Name and verbalise meaningful relationships.
6. Define multiplicities and constraints. Capture structural business rules.
7. Investigate difficult relationships. Look for hidden classes and roles.
8. Use generalisation and whole-part structures carefully.
9. Cross-check processing and information. Use CRUD and scenarios.
10. Iterate. Let new requirements change the model.

FINAL THOUGHT

The class diagram is not something you draw after requirements analysis. Building it is part of requirements analysis.

Course complete

Look beyond the verb. Ask what the noun means, what must be known about it, and how it relates to everything else.