

UNIT 4

Design

(12 Hours) — System Analysis and Design (CACS203)

1. Designing Databases

1.1 Introduction and Database Design

In the design phase the conceptual data model produced in analysis is converted into a **logical** and then a **physical** database design. A **database** is an organized collection of logically related data; a **DBMS** (e.g., MySQL, PostgreSQL, Oracle, SQL Server) manages it. Good design reduces redundancy, protects integrity, speeds retrieval and supports change. The steps are:

- **Conceptual design** — ER model (from analysis).
- **Logical design** — relations, keys, normalization (independent of DBMS).
- **Physical design** — data types, indexes, file organization, storage (specific to DBMS).

1.2 The Relational Database Model

A **relation** is a named, two-dimensional table of data with rows (**tuples**) and columns (**attributes**). Properties of a relation: each cell holds a single value; each column has a unique name and a domain; each row is unique; order of rows and columns is irrelevant.

Term	Meaning
Primary key	Attribute(s) that uniquely identify each row; cannot be null
Candidate key	Any attribute set that could serve as a primary key
Composite key	Primary key made of two or more attributes
Foreign key	Attribute in one relation that matches the primary key of another, creating a relationship
Entity integrity	No primary-key attribute may be null
Referential integrity	A foreign key value must match an existing primary key value or be null

Relations are written in shorthand: **STUDENT**(StudentID, Name, Phone, DeptID) — underline = primary key, italic = foreign key.

1.3 Normalization

Normalization is the process of decomposing relations to remove anomalies caused by redundancy. **Insertion, deletion and modification anomalies** arise when unrelated facts are stored in one relation. Normalization is based on **functional dependencies** ($A \rightarrow B$: value of A determines B).

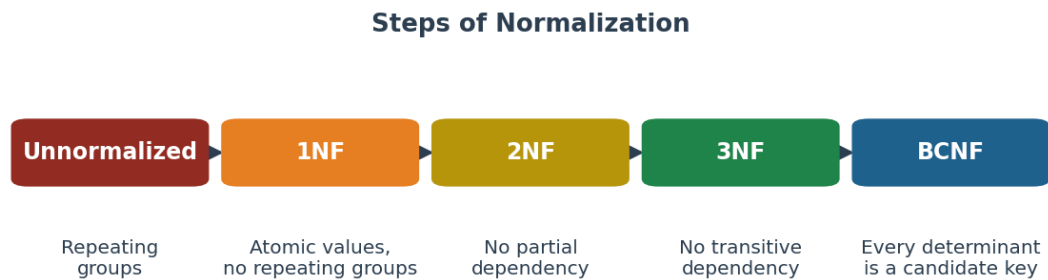


Figure 4.1 — Steps of normalization

Normal form	Rule	Fix
First (1NF)	All attributes atomic, no repeating groups, has a primary key	Split repeating groups into rows / new relation
Second (2NF)	1NF and no partial dependency: non-key attributes depend on the whole key	Move partially dependent attributes to a new relation
Third (3NF)	2NF and no transitive dependency (non-key $\rightarrow$ non-key)	Move attributes dependent on a non-key to a new relation
BCNF	Every determinant is a candidate key	Decompose on the offending determinant

Example 4.1 — Normalize an order relation

ORDER(OrderID, OrderDate, CustID, CustName, CustCity, {ProductID, ProductName, UnitPrice, Qty}) where the braces mean a repeating group. Normalize to 3NF.

Solution

1NF: remove the repeating group: ORDER(OrderID, OrderDate, CustID, CustName, CustCity) and ORDER_LINE(OrderID, ProductID, ProductName, UnitPrice, Qty).

2NF: ProductName and UnitPrice depend only on ProductID (part of the key) → PRODUCT(ProductID, ProductName, UnitPrice); ORDER_LINE(OrderID, ProductID, Qty).

3NF: CustName and CustCity depend on CustID, not on OrderID (transitive) → CUSTOMER(CustID, CustName, CustCity); ORDER(OrderID, OrderDate, *CustID*).

Final: ORDER, ORDER_LINE, PRODUCT, CUSTOMER — all in 3NF.

1.4 Transforming E-R Diagrams into Relations

E-R construct	Rule for the relational model
Regular entity	Becomes a relation; identifier → primary key; simple attributes → columns; composite → its components
Weak entity	Relation with the owner's primary key as foreign key; primary key = owner key + partial key
Multivalued attribute	New relation with the entity's key plus the attribute value
Binary 1:M relationship	Place the primary key of the “one” side as a foreign key in the “many” side
Binary M:N relationship	New (associative) relation whose primary key combines both entity keys, plus relationship attributes
Binary 1:1 relationship	Put the primary key of one relation as a foreign key in the other (usually the one with mandatory participation)
Unary relationship	Foreign key referencing the same relation (1:M) or a new relation (M:N)
Supertype/subtype	Relation for supertype and each subtype; subtype key = supertype key

Example 4.2 — Map the Student–Course ERD

STUDENT(StudentID, Name) enrolls in COURSE(CourseID, Title) as M:N with attribute Grade.

Solution

STUDENT(StudentID, Name)

COURSE(CourseID, Title)

ENROLLMENT(StudentID, CourseID, Grade) — StudentID and CourseID are also foreign keys.

1.5 Merging Relations

When designs from different user views are combined, relations are **merged** to form one integrated set. Problems to resolve: **synonyms** (different names for the same attribute, e.g., CustNo and ClientID), **homonyms** (same name for different things), **transitive dependencies** introduced by merging, and **supertype/subtype** relationships. After merging, the relations are checked again for 3NF.

1.6 Physical File and Database Design

Physical design turns 3NF relations into technical specifications. Inputs are the normalized relations, volume estimates, attribute definitions, response time and security needs, and DBMS features. Decisions are made about **data types**, **file organization**, **indexes**, and **denormalization**.

1.7 Designing Fields

A **field** is the smallest unit of named application data. Choose a **data type** that represents the values, saves space, improves integrity and supports needed operations (e.g., INTEGER, DECIMAL, VARCHAR, DATE, BOOLEAN). Integrity controls include:

- **Default value**, **range** and **referential** control, **null value** control.
- **Format / mask** and **coding** techniques (e.g., codes M/F, compression).
- Handling **missing data**: substitute default, trigger a warning, or allow null.
- **Calculated fields** are often not stored unless required for speed.

1.8 Designing Physical Tables

A physical table corresponds to a relation. To improve performance, designers may apply:

- **Denormalization** — combining relations or duplicating data to avoid costly joins when read speed matters (costs storage and integrity).
- **Partitioning** — splitting a table horizontally (rows) or vertically (columns).
- **Indexes** — secondary structures (B-tree, hash) on frequently searched, joined or sorted fields; too many indexes slow updates.
- **File organization** — **sequential** (sorted; efficient for batch), **indexed** (fast direct and sequential access), **hashed** (fastest direct access on key).

Example 4.3 — Should you index?

A table STUDENT has 1,000,000 rows. Queries frequently search by Roll_No and by Email; the table receives few updates. Suggest indexing.

Solution

Create indexes on **Roll_No** (primary key) and **Email** (unique) — searches change from scanning up to 1,000,000 rows to a few index look-ups; the overhead on infrequent updates is small.

2. Designing Forms and Reports

2.1 Introduction and Design

Forms are business documents containing predefined data with spaces for entering data (e.g., admission form, order screen). **Reports** are business documents containing only predefined data, produced for reading (e.g., invoice, marks sheet, sales summary). Both are the most visible part of a system, so they strongly affect users' opinion of it.

Form/report type	Purpose
Data-entry form	Capture new data or edit existing data
Detail report	Lists every record in a file (e.g., full sales list)
Summary report	Aggregates data (totals, averages)
Exception report	Shows only records that violate a condition (e.g., low stock)
Scheduled / on-demand report	Produced at fixed intervals or when requested

The design process: (1) gather requirements — who, what, when, where, how many; (2) prototype layouts on paper, in a design tool or in a RAD tool; (3) user review; (4) build and test.

2.2 Formatting Forms and Reports

- **Meaningful titles** — clear, specific; include version, date and page number for reports.
- **Meaningful information** — only what is needed; use the same sequence as source documents.
- **Balanced layout** — logical grouping, consistent alignment, adequate white space, related fields together.
- **Easy navigation** — logical tab order, page numbers, clear labels, top-to-bottom, left-to-right flow.

- **Field labels and formats** — right-align numbers, left-align text, use consistent date and currency formats; mark required fields.
- **Use of colour and highlighting** sparingly (bold, reverse video, blinking only for critical information); ensure sufficient contrast and do not rely on colour alone.
- **Display of text** — mixed-case, short sentences, consistent abbreviations, avoid jargon.
- **Tables and lists** — column headings, group by category, subtotals, sort meaningfully.

Figure 4.2 — Sample data-entry form layout

2.3 Assessing Usability

Usability is the ease with which a system can be learned and used. It is measured by:

Factor	Question
Effectiveness	Can users accomplish their tasks correctly?
Efficiency	How quickly, with how few steps and errors?
Learnability	How quickly can a new user become productive?
Satisfaction	Do users find it pleasant, comfortable?
Memorability / Error tolerance	Can users recall it after a break? Can they recover from mistakes?

Usability is assessed by **usability testing** (observing users on real tasks), **heuristic evaluation** by experts, questionnaires (e.g., SUS) and analysis of error logs. The cost of poor usability includes lost productivity, errors, training cost and abandoned systems.

3. Designing Interfaces and Dialogues

3.1 Introduction and Design

A **user interface** is the point of interaction between users and the system. A **dialogue** is the sequence of interaction between the user and the system. Design principles (Nielsen, Shneiderman): **consistency, visibility of system status, user control and freedom, error prevention and helpful error messages, recognition rather than recall, flexibility for novices and experts, minimalist design, and feedback** for every action.

3.2 Interaction Methods and Devices

Method	Description	Best for
Command language	User types commands	Expert users, scripting
Menu	Choose from a list of options (menu bar, pop-up, drop-down)	Novice/occasional users
Form interaction	Fill in fields on a screen	Data entry
Object-based (GUI)	Manipulate icons, windows, pointers (WIMP)	General users
Natural language / voice	Speak or type in plain language	Hands-free, assistants
Touch / gesture	Tap, swipe, pinch on touchscreen	Mobile devices, kiosks

Input devices: keyboard, mouse, touchscreen, stylus, scanner, barcode/QR reader, microphone, camera and biometric sensors. **Output devices:** monitor, printer, speaker and haptic feedback. Selection depends on user, task, environment, accuracy and cost.

3.3 Designing Interfaces and Dialogues in Graphical Environments

Graphical user interfaces present **windows, icons, menus and pointers** and controls such as text boxes, radio buttons (single choice), check boxes (multiple choice), list/combo boxes and buttons.

- Design the **dialogue sequence** first, using a **dialogue diagram** showing the screens/forms and the transitions between them.
- Provide **feedback**: progress bars, confirmations, and clear **error messages** that say what went wrong and how to fix it.
- Support **help**: tool-tips, context-sensitive help, examples in fields.
- Ensure **consistency** across screens (positions of buttons, colours, terms) and follow platform guidelines.

- Design for **accessibility** (contrast, keyboard access, screen-reader labels) and **responsive** layouts for different screen sizes.
- Provide **undo** and confirmations for destructive actions like “Delete”.

Example 4.4 — Choose the control

Which control is best for (a) selecting gender, (b) choosing hobbies, (c) selecting a country from 200, (d) entering a password?

Solution

- (a) **Radio buttons** — one exclusive choice from few options.
- (b) **Check boxes** — multiple choices.
- (c) **Drop-down / combo box with search** — many options in small space.
- (d) **Password text box** with masked characters and, optionally, a show/hide toggle.

Key Takeaways

- ✓ Design converts the conceptual model into normalized relations (3NF/BCNF) and then a physical database with fields, indexes and tables.
- ✓ E-R diagrams map to relations: entities to tables, 1:M via foreign keys, M:N via associative tables.
- ✓ Forms and reports should be clear, consistent, well laid out and evaluated for usability.
- ✓ Interfaces and dialogues use menus, forms, GUI and touch; follow consistency, feedback and error-prevention principles.
- ✓ Denormalization and indexing are performance trade-offs applied at the physical design stage.

— End of Unit 4 —