



BASICS OF DATABASE MANAGEMENT SYSTEMS

ITT06104

DIT2 - MARCH INTAKE 2022/2023

COLLEGE OF BUSINESS EDUCATION

MR. KIKWEMBO MLEKWA

Course Summary

- ▶ Information Gathering Techniques
- ▶ Comparison of Traditional File System and Database Approach
- ▶ Data Models in Relational Database
- ▶ Introduction to SQL
- ▶ Management and Security of Data
- ▶ Transactions Concept in Database Management Systems

Information Gathering Techniques

LECTURE ONE

Information Engineering Planning Phase

Identify Strategic Planning Factors

- Goals
- Critical success factors
- Problems areas

Identify Corporate Planning Object

- Organizational Units
- Locations
- Business Functions
- Entity Types

Information Engineering Planning Phase

Develop an Enterprise Model

- Functional decomposition
- Entity-relationship diagram
- Planning matrices.

Planning for Database Development

Enterprise Modelling

- Analyse current data processing
- Analyse the general business functions and their database needs
- Justify need for new data and databases in support of business

Planning for Database Development

Conceptual Data Modelling

- Identify scope of database requirements for proposed information system
- Analyse overall data requirements for business function supported by database.
- Develop preliminary conceptual data model, including entities and relationships.

Information Gathering

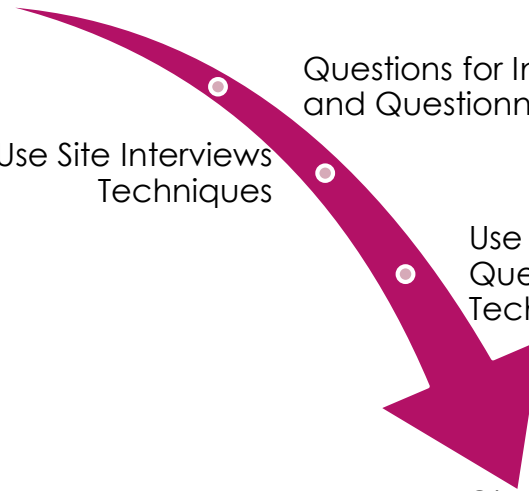
Techniques for
Information
gathering

Questions for Interview
and Questionnaires

Use Site Interviews
Techniques

Use
Questionnaire
Technique

Use Observation
Technique



Techniques for Information Gathering



Questionnaire

Interviews

Observation

Questionnaire Technique

INFORMATION GATHERING
TECHNIQUES

Questionnaire

- ▶ A **questionnaire** is a list of questions or items used to gather data from respondents about their attitudes, experiences, or opinions. Questionnaires can be used to collect quantitative and/or qualitative information.
- ▶ Designing a questionnaire means creating valid and reliable questions that address your research objectives, placing them in a useful order, and selecting an appropriate method for administration.

Questionnaire

- ▶ A questionnaire is a research tool featuring a series of questions used to collect useful information from respondents.
- ▶ These instruments include either written or oral questions and comprise an interview-style format.
- ▶ Questionnaires may be qualitative or quantitative and can be conducted online, by phone, on paper or face-to-face, and questions don't necessarily have to be administered with a researcher present.

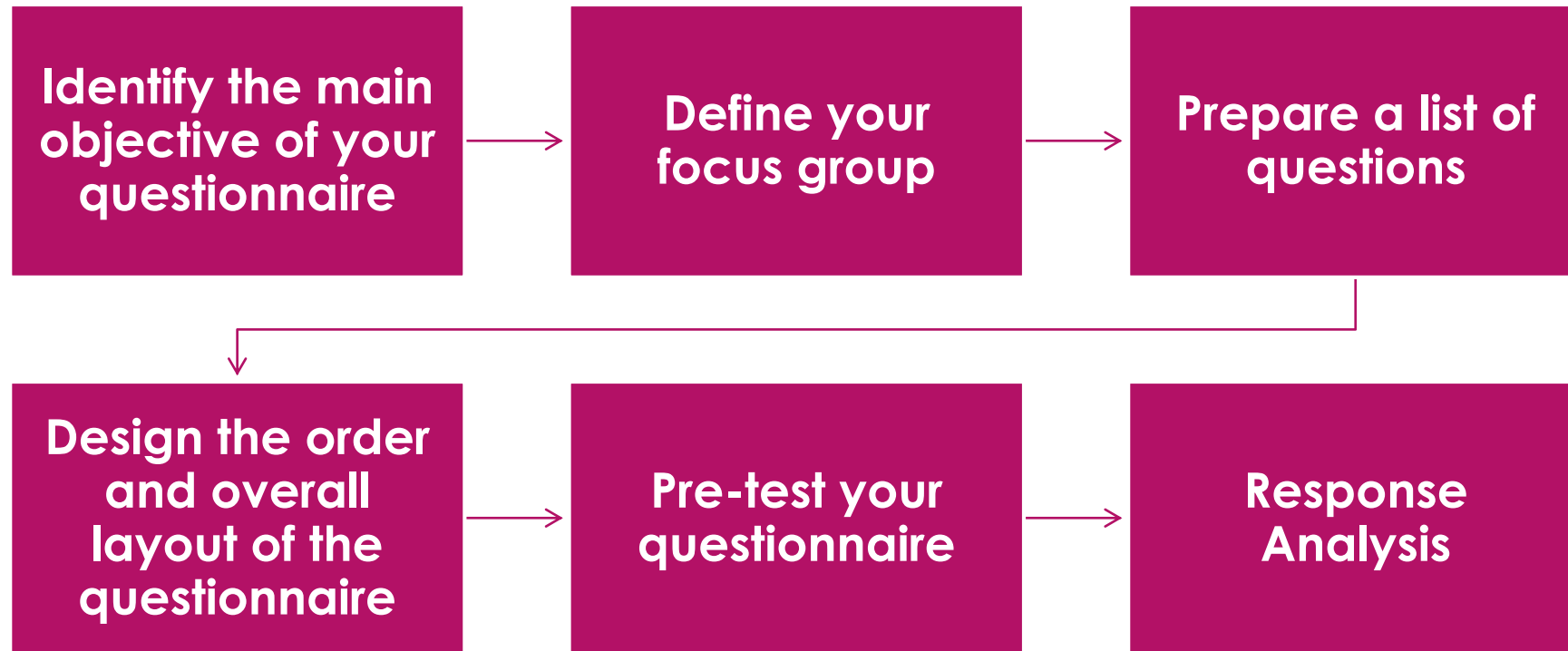
Questionnaire

- ▶ Questionnaires feature either open or closed questions and sometimes employ a mixture of both.
- ▶ Open-ended questions enable respondents to answer in their own words in as much or as little detail as they desire.
- ▶ Closed questions provide respondents with a series of predetermined responses they can choose from.

Questionnaire

- ▶ A questionnaire is a research tool of sampling that consists of a set of questions prepared for a target audience.
- ▶ Sometimes it becomes necessary to research a particular topic where you want to conclude.
- ▶ You share this set of questions with the respondents.
- ▶ The purpose of a questionnaire can be to gain feedback, understand opinions, or collect data for research within a period

Steps for Preparation of Questions



Tips for a good Questionnaire

- ▶ Keep in mind the underlying assumption
- ▶ The format of the questions depends on the method
- ▶ Avoid open-ended questions if possible
- ▶ Write down a research question and focus group
- ▶ One or more time frames
- ▶ Try to minimize the number of questions
- ▶ Add some questions to capture attributes
- ▶ Do not use the 'Other' category
- ▶ Put open-ended questions first
- ▶ Always pre-test

Site Interviews Technique

INFORMATION GATHERING
TECHNIQUES

Site Interview Techniques

- ▶ Interviews can be defined as a qualitative research technique which involves “conducting intensive individual interviews with a small number of respondents to explore their perspectives on a particular idea, program or situation.

Interviews

Structured Interviews

Unstructured Interviews

Semi-structured Interviews

Interviews

Structured Interviews

- consist of a series of pre-determined questions that all interviewees answer in the same order.
- Data analysis usually tends to be more straightforward because researcher can compare and contrast different answers given to the same questions.

Interviews

Unstructured Interviews

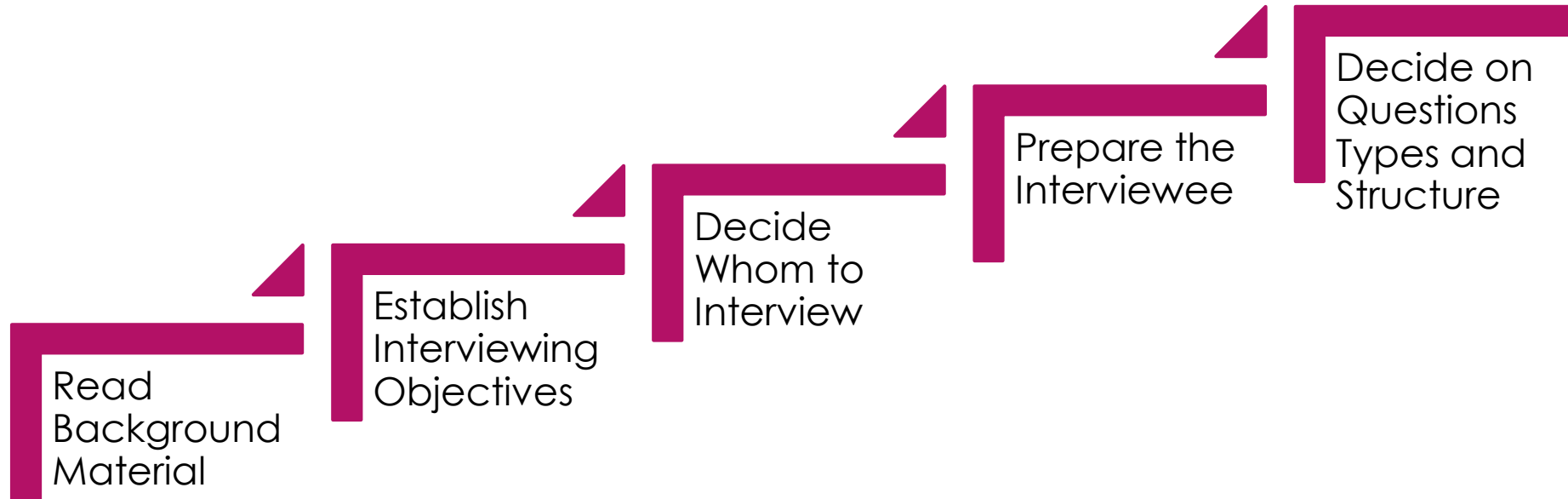
- are usually the least reliable from research viewpoint, because no questions are prepared prior to the interview and data collection is conducted in an informal manner.
- Unstructured interviews can be associated with a high level of bias and comparison of answers given by different respondents tends to be difficult due to the differences in formulation of questions.

Interviews

Semi-Structured Interviews

- contain the components of both, structured and unstructured interviews.
- In semi-structured interviews, interviewer prepares a set of same questions to be answered by all interviewees.
- At the same time, additional questions might be asked during interviews to clarify and/or further expand certain issues.

Preparation for Interview Questions



Observation Technique

INFORMATION GATHERING
TECHNIQUES

Observation Method

- ▶ Observation, as the name implies, is a way of collecting data through observing.
- ▶ This data collection method is classified as a participatory study, because the researcher has to immerse herself in the setting where her respondents are, while taking notes and/or recording.
- ▶ Observation data collection method may involve watching, listening, reading, touching, and recording behavior and characteristics of phenomena.

Observation Method

- ▶ Advantages of observation data collection method include direct access to research phenomena,
- ▶ high levels of flexibility in terms of application and generating a permanent record of phenomena to be referred to later.
- ▶ At the same time, this method is disadvantaged with longer time requirements, high levels of observer bias, and impact of observer on primary data, in a way that presence of observer may influence the behavior of sample group elements.

Example of Questionnaire and Interview Questions

INFORMATION GATHERING

Sample Questions

What Are the Strategic Goals and Expectations of the Software Application for Your Business?

- Understanding the value of the future product for customers and the target audience starts with knowing the problem it solves.
- Some apps have clearly defined functional goals (process automation, business problem solving), and some serve to gain a competitive advantage or are of an image-building nature.

Sample Questions

What Is Your Software Development Budget?

- Getting down to software development without knowing the deadline set and the budget available is a completely lame idea
- Any company has certain financial restrictions in addition to which some contingencies always appear in the course of work.
- If you skip taking into account these aspects, customers may run out of money at a point when they have already spent a lot of resources on development, and the product release is still a long haul ahead.

Sample Questions

What Are We Delivering and What Are We Not?

- Any product is a complex of different elements. The work on a project includes various stages that we have to complete in a specific order.
- For instance, website development includes work in the following areas:
- Programming, design creation, content creation, design template layout, or testing.

Sample Questions

What Is the List of the Functionalities You Want to Implement in the Future?

- This question is important for making the right architectural decisions.
- After all, you may envision your product in several future years and understand what capabilities you will have to implement at that time.
- Taking everything into account, you can develop an excellent evolution roadmap for your product.

Sample Questions

What is the Core Functionality of the Product?

- Never try doing everything at once.
- That is one of the most common reasons leading to project failure, as customers may just lack time and money to complete it due to unforeseen circumstances.
- You may plan as many features as you like, but first of all, you have to define a small set of them to be available in the initial app release

Sample Questions

What are the Requirements for Data Storing?

- While operating, any company has to collect, store, or process data.
- What kind of data it will be (texts, photos, videos, etc.), what storage rules will apply to it, how it will get processed and, if necessary, deleted — you have to know the answers to these questions before you start the development process.
- This knowledge will allow you to select the optimal database storage solution.

Data Models in Relational Database

RELATIONAL DATABASE

Relationship

- ▶ In relational databases, a relationship exists between two tables when one of them has a foreign key that references the primary key of the other table.
- ▶ This single fact allows relational databases to split and store data in different tables, yet still link the disparate data items together.

Data Models in Relational Database

- ▶ Data models are visual representations of an enterprise's data elements and the connections between them.
- ▶ By helping to define and structure data in the context of relevant business processes, models support the development of effective information systems.

Data Models

- ▶ Data models are visual representations of an enterprise's data elements and the connections between them.
- ▶ By helping to define and structure data in the context of relevant business processes, models support the development of effective information systems.
- ▶ They enable business and technical resources to collaboratively decide how data will be stored, accessed, shared, updated and leveraged across an organization.

Data Models

- ▶ **Conceptual data models**

- ▶ Also known as domain models, conceptual data models explore and detail your high-level, static business structures and concepts.
- ▶ They are most frequently used during the beginning of a new project, when high-level concepts and initial requirements are hashed out.
- ▶ Often, they are created as precursors or alternatives to the next stage: logical data models

Data Models

- ▶ **Logical data models**

- ▶ After your problem domain and initial concepts become more clear through conceptual data modeling, it's time to get more specific with a logical data model.
- ▶ Whether you're looking through the lens of a single project or your entire enterprise, these models clarify the various logical entities (types or classes of data) you'll be working with, the data attributes that define those entities, and the relationships between them.

Data Models

► **Physical data models**

- When you get to the physical data modeling stage, it's truly time to get down to the nitty-gritty.
- These models are used to design the internal schema of a database.
- That includes all of the various tables, the columns on those tables and the relationships between them.
- These models will be directly translated into production database design, which will support further development of information systems.
- Physical data models generally are used to design three types of databases: relational for traditional operational databases, document for NoSQL and JSON databases, and dimensional for aggregation and business intelligence data stores such as data warehouses and data marts.

Entity in a Database

- ▶ An entity in a database table is defined with the 'fixed' set of attributes.
- ▶ For example, if we have to define a student entity then we can define it with the set of attributes like roll number, name, course.
- ▶ The attribute values, of each student entity, will define its characteristics in the table.
- ▶ An entity can be a real-world object, either animate or inanimate, that can be easily identifiable.

Entity

- ▶ For example, in a school database, students, teachers, classes, and courses offered can be considered as entities.
- ▶ All these entities have some attributes or properties that give them their identity.
- ▶ An entity set is a collection of similar types of entities.
- ▶ An entity set may contain entities with attribute sharing similar values.
- ▶ For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.



Entity

A single unique object in the real world that is being mastered. Examples of an entity are a single person, single product, or single organization.

Entity type

A person, organization, object type, or concept about which information is stored. Describes the type of the information that is being mastered. An entity type typically corresponds to one or several related tables in database.



Attribute

A characteristic or trait of an entity type that describes the entity, for example, the Person entity type has the Date of Birth attribute.

Record

The storage representation of a row of data.

Member record

The representation of the entity as it is stored in individual source systems. Information for each member record is stored as a single record or a group of records across related database tables

ERD

- ▶ An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology (IT) system.
- ▶ An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how “entities” such as people, objects or concepts relate to each other within a system.

ERD

- ▶ ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research.
- ▶ Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.

ProductFeature
name: string
description: string
category: string

ProductFeature
name: string
description: string
category: string

ProductFeature
name: string

ProductFeature
id: string
name: string
description: string
category: string
parentFeatureId: string
childrenFeatureIds: string[]
isActive: boolean

ProductFeature
id: string
name: string
category: string

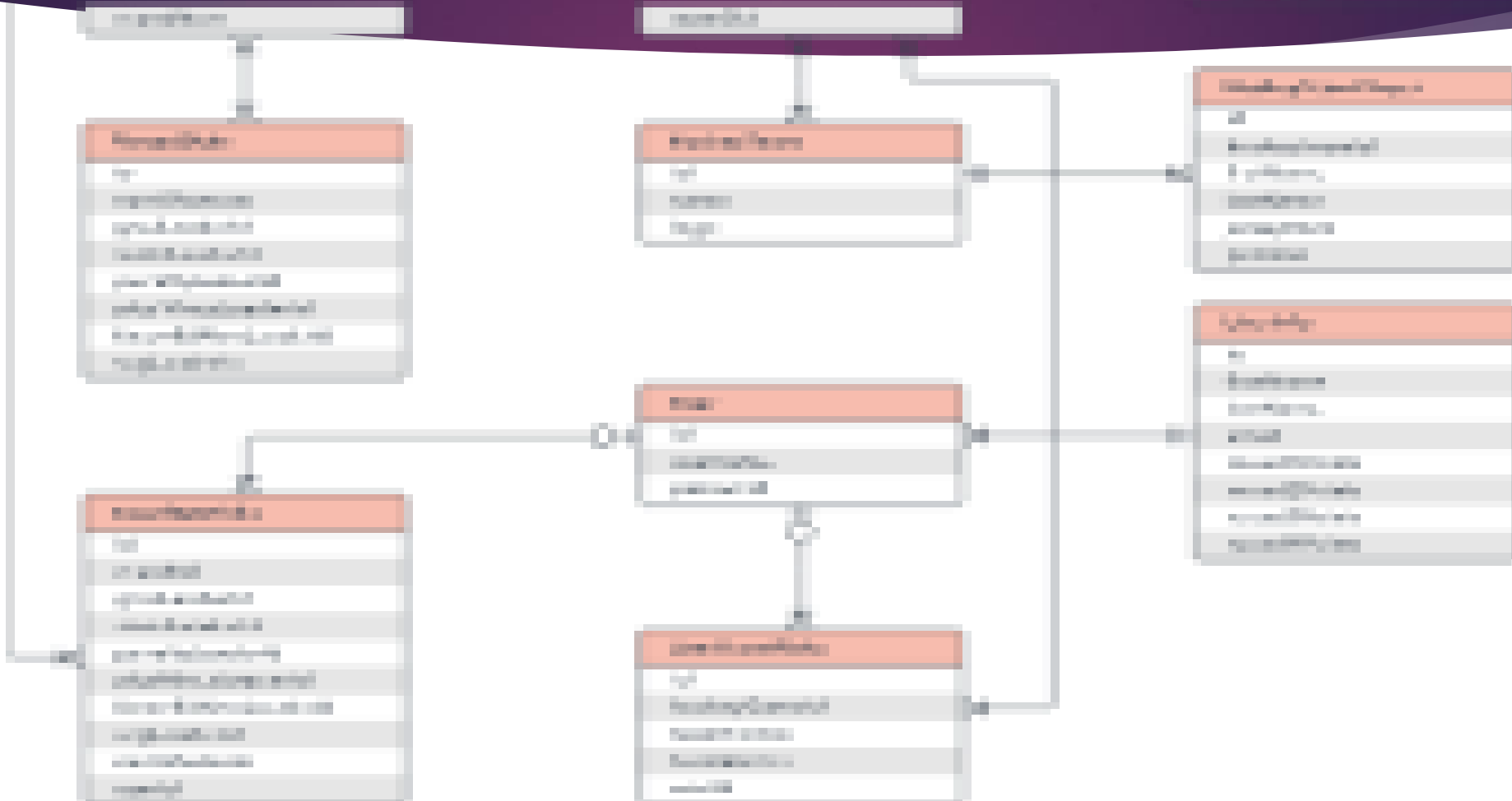
ProductFeature
id: string
name: string
description: string
category: string
parentFeatureId: string

ProductFeature
id: string
name: string
description: string
category: string
parentFeatureId: string
childrenFeatureIds: string[]
isActive: boolean
createdAt: string
updatedAt: string
deletedAt: string

ProductFeature
id: string
name: string
category: string

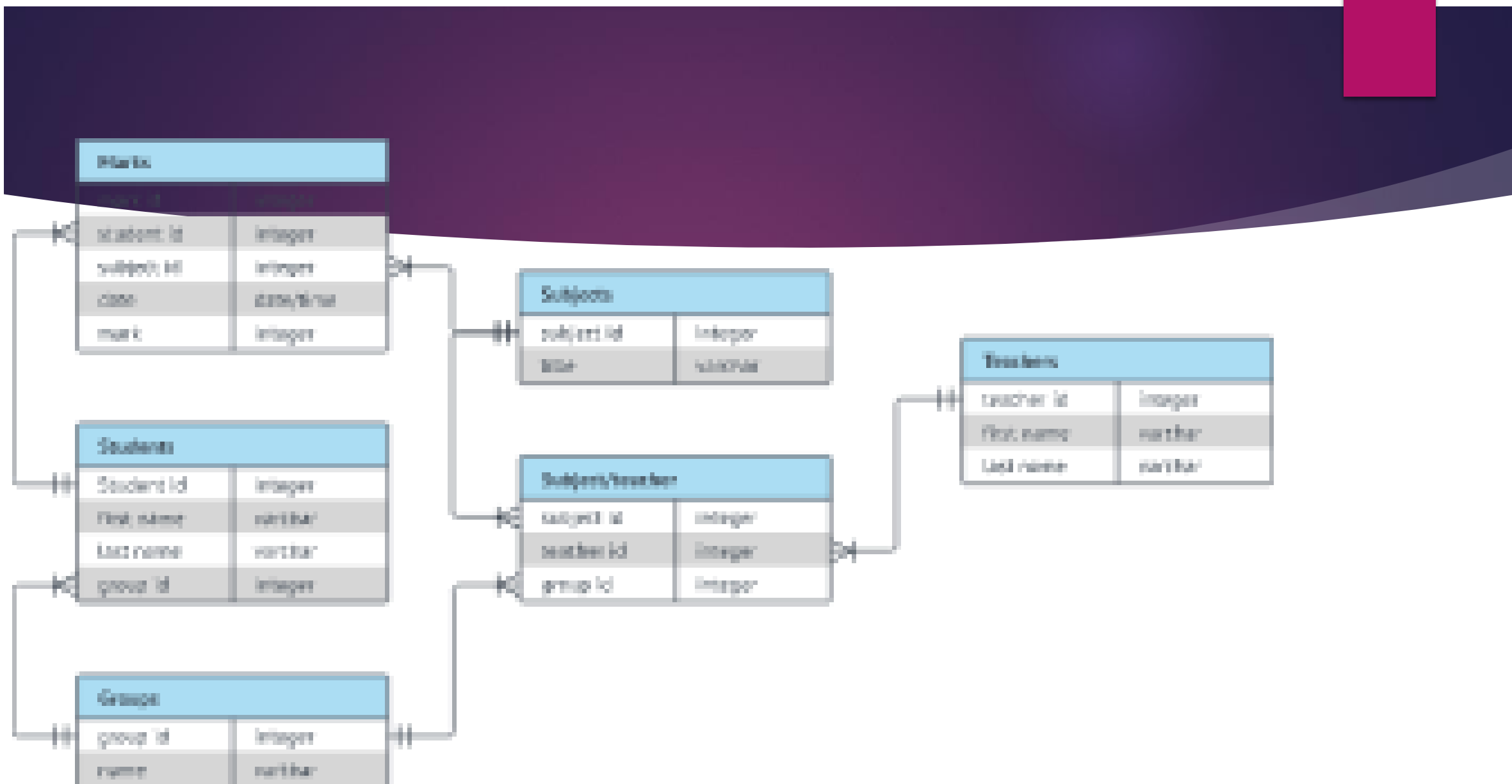
ProductFeature
id: string
name: string
description: string
category: string
parentFeatureId: string
childrenFeatureIds: string[]

ProductFeature
id: string
name: string
description: string
category: string



ERD

- ▶ ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves.
- ▶ ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.



Natural Language to ERD

- ▶ **Mapping natural language**
- ▶ ER components can be equated to parts of speech, as Peter Chen did. This shows how an ER Diagram compares to a grammar diagram:
- ▶ Common noun: Entity type. Example: student.
- ▶ Proper noun: Entity. Example: Sally Smith.
- ▶ Verb: Relationship type. Example: Enrolls. (Such as in a course, which would be another entity type.)
- ▶ Adjective: Attribute for entity. Example: sophomore.
- ▶ Adverb: Attribute for relationship. Example: digitally.
- ▶ The database query language ERROL actually mimics natural language constructs. ERROL is based on reshaped relational algebra (RRA) and works with ER models, capturing their linguistic aspects.

How to draw a basic ER diagram

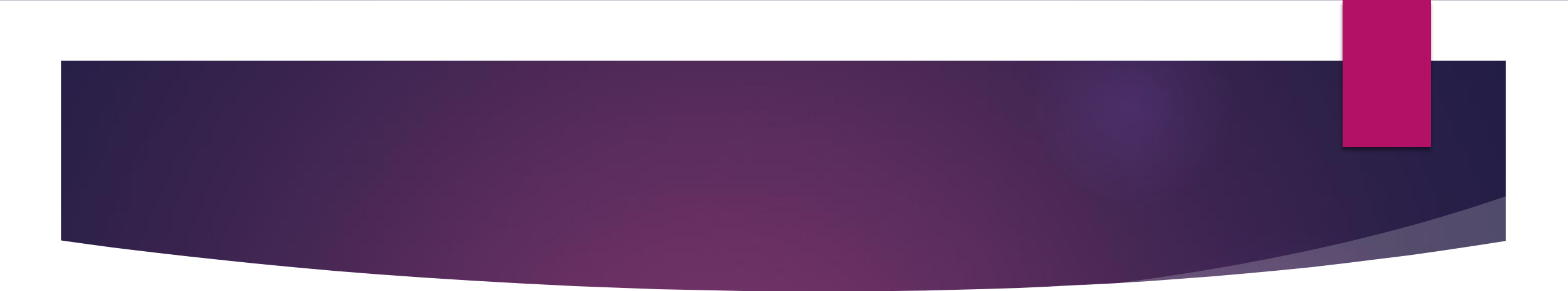
- ▶ Purpose and scope: Define the purpose and scope of what you're analyzing or modeling.
- ▶ Entities: Identify the entities that are involved. When you're ready, start drawing them in rectangles (or your system's choice of shape) and labeling them as nouns.
- ▶ Relationships: Determine how the entities are all related. Draw lines between them to signify the relationships and label them. Some entities may not be related, and that's fine. In different notation systems, the relationship could be labeled in a diamond, another rectangle or directly on top of the connecting line.
- ▶ Attributes: Layer in more detail by adding key attributes of entities. Attributes are often shown as ovals.
- ▶ Cardinality: Show whether the relationship is 1-1, 1-many or many-to-many.

NORMALIZATION

- ▶ In simple terms, data normalization is the practice of organizing data entries to ensure they appear similar across all fields and records, making information easier to find, group and analyze.
- ▶ **Normalization** is a database design technique that reduces data redundancy and eliminates undesirable characteristics like Insertion, Update and Deletion Anomalies.
- ▶ Normalization rules divides larger tables into smaller tables and links them using relationships.
- ▶ The purpose of Normalisation in SQL is to eliminate redundant (repetitive) data and ensure data is stored logically.

NORMALIZATION

- ▶ If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator.
- ▶ Managing a database with anomalies is next to impossible.
- ▶ **Update anomalies** – If data items are scattered and are not linked to each other properly, then it could lead to strange situations.
- ▶ For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values.
- ▶ Such instances leave the database in an inconsistent state.

- 
- ▶ **Deletion anomalies** – We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
 - ▶ **Insert anomalies** – We tried to insert data in a record that does not exist at all.
 - ▶ Normalization is a method to remove all these anomalies and bring the database to a consistent state.

NORMALIZATION


- ▶ The inventor of the relational model Edgar Codd proposed the theory of normalization of data with the introduction of the First Normal Form, and he continued to extend theory with Second and Third Normal Form. Later he joined Raymond F. Boyce to develop the theory of Boyce-Codd Normal Form.

Database Normal Forms

- ▶ Here is a list of Normal Forms in SQL:
- ▶ 1NF (First Normal Form)
- ▶ 2NF (Second Normal Form)
- ▶ 3NF (Third Normal Form)
- ▶ BCNF (Boyce-Codd Normal Form)
- ▶ 4NF (Fourth Normal Form)
- ▶ 5NF (Fifth Normal Form)
- ▶ 6NF (Sixth Normal Form)

NORMALIZATION – 1NF

- ▶ First Normal Form (1NF)
- ▶ The most basic form of data normalization is 1NFm which ensures there are no repeating entries in a group.
- ▶ To be considered 1NF, each entry must have only one single value for each cell and each record must be unique.
- ▶ For example, you are recording the name, address, gender of a person, and if they bought cookies.



Course	Content
Programming	Java, c++
Web	HTML, PHP, ASP

We re-arrange the relation (table) as below, to convert it to First Normal Form.

Course	Content
Programming	Java
Programming	C++
Web	HTML
Web	PHP
Web	ASP

NORMALIZATION – 2NF

- ▶ Second Normal Form (2NF)
- ▶ Again working to ensure no repeating entries, to be in the 2NF rule, the data must first apply to all the 1NF requirements.
- ▶ Following that, data must have only one primary key.
- ▶ To separate data to only have one primary key, all subsets of data that can be placed in multiple rows should be placed in separate tables.

NORMALIZATION – 2NF

- ▶ Then, relationships can be created through new foreign key labels.
- ▶ For example, you are recording the name, address, gender of a person, if they bought cookies, as well as the cookie types.
- ▶ The cookie types are placed into a different table with a corresponding foreign key to each person's name.

NORMALIZATION – 2NF

- ▶ Before we learn about the second normal form, we need to understand the following –
- ▶ **Prime attribute** – An attribute, which is a part of the candidate-key, is known as a prime attribute.
- ▶ **Non-prime attribute** – An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

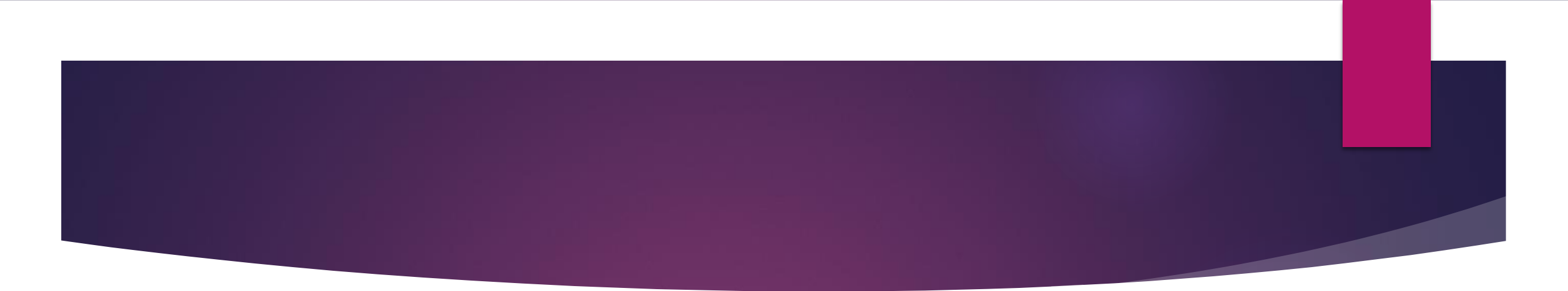
NORMALIZATION – 2NF

- ▶ If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if $X \rightarrow A$ holds, then there should not be any proper subset Y of X , for which $Y \rightarrow A$ also holds true.

Student_Project

Stu_ID	Proj_ID	Stu_Name	Proj_Name
--------	---------	----------	-----------



- 
- ▶ We see here in Student_Project relation that the prime key attributes are Stu_ID and Proj_ID.
 - ▶ According to the rule, non-key attributes, i.e. Stu_Name and Proj_Name must be dependent upon both and not on any of the prime key attribute individually.
 - ▶ But we find that Stu_Name can be identified by Stu_ID and Proj_Name can be identified by Proj_ID independently.
 - ▶ This is called **partial dependency**, which is not allowed in Second Normal Form.



Student

Stu_ID	Stu_Name	Proj_ID
--------	----------	---------

Project

Proj_ID	Proj_Name
---------	-----------

NORMALIZATION – 3NF

- ▶ Third Normal Form (3NF)
- ▶ For data to be in this rule, it must first comply with all the 2NF requirements. Following that, data in a table must only be dependent on the primary key.
- ▶ If the primary key is changed, all data that is impacted must be put into a new table.
- ▶ For example, you are recording the name, address, and gender of a person but go back and change the name of a person..

NORMALIZATION – 3NF

- ▶ When you do this, the gender may then change as well.
- ▶ To avoid this, in 3NF gender is given a foreign key and a new table to store gender.
- ▶ As you begin to better understand the normalization forms, the rules will become more clear while separating your data into tables and levels will become effortless.
- ▶ These tables will then make it simple for anyone within an organization to gather information and ensure they collect correct data that is not duplicated

Student_Detail

Stu_ID

Stu_Name

City

Zip



- ▶ We find that in the above Student_detail relation, Stu_ID is the key and only prime key attribute.
- ▶ We find that City can be identified by Stu_ID as well as Zip itself. Neither Zip is a superkey nor is City a prime attribute.
- ▶ Additionally, $\text{Stu_ID} \rightarrow \text{Zip} \rightarrow \text{City}$, so there exists **transitive dependency**.



Student_Detail

Stu_ID	Stu_Name	Zip
--------	----------	-----

ZipCodes

Zip	City
-----	------

TYPES OF KEYS IN DBMS

- ▶ **What are the different types of Keys in DBMS?**
- ▶ Keys are of seven broad types in DBMS:
- ▶ Candidate Key
- ▶ Primary Key
- ▶ Foreign Key
- ▶ Super Key
- ▶ Alternate Key
- ▶ Composite Key
- ▶ Unique Key

TYPES OF KEYS IN DBMS

► **Primary Key**

- The primary key refers to a column or a set of columns of a table that helps us identify all the records uniquely present in that table. A table can consist of just one primary key. Also, this primary key cannot consist of the same values reappearing/repeating for any of its rows. All the values of a primary key have to be different, and there should be no repetitions.
- The PK (PRIMARY KEY) constraint that we put on a column/set of columns won't allow these to have a null value or a duplicate. Any table can consist of only a single primary key constraint. A foreign key (explained below) that refers to it can never change the values present in the primary key.

TYPES OF KEYS IN DBMS

► **Super Key**

- A super key refers to the set of all those keys that help us uniquely identify all the rows present in a table. It means that all of these columns present in a table that can identify the columns of that table uniquely act as the super keys.
- A super key is a candidate key's superset (candidate key has been explained below). We need to pick the primary key of any table from the super key's set so as to make it the table's identity attribute.

TYPES OF KEYS IN DBMS

- ▶ **Candidate Key**

- ▶ The candidate keys refer to those attributes that identify rows uniquely in a table. In a table, we select the primary key from a candidate key. Thus, a candidate key has similar properties as that of the primary keys that we have explained above. In a table, there can be multiple candidate keys.

TYPES OF KEYS IN DBMS

- ▶ **Alternate Key**

- ▶ As we have stated above, any table can consist of multiple choices for the primary key. But, it can only choose one. Thus, all those keys that did not become a primary key are known as alternate keys.

TYPES OF KEYS IN DBMS

- ▶ **Foreign Key**

- ▶ We use a foreign key to establish relationships between two available tables. The foreign key would require every value present in a column/set of columns to match the referential table's primary key. A foreign key helps us to maintain data as well as referential integrity.

TYPES OF KEYS IN DBMS

- ▶ **Composite Key**

- ▶ The composite key refers to a set of multiple attributes that help us uniquely identify every tuple present in a table. The attributes present in a set may not be unique whenever we consider them separately. Thus, when we take them all together, it will ensure total uniqueness.

CONSTRAINTS

- ▶ constraints are guidelines or limitations imposed on database tables to maintain the integrity, correctness, and consistency of the data. Constraints can be used to enforce data linkages across tables, verify that data is unique, and stop the insertion of erroneous data

CONSTRAINTS

- ▶ **The following can be guaranteed via constraints**
- ▶ **Data Accuracy** – Data accuracy is guaranteed by constraints, which make sure that only true data is entered into a database.
- ▶ For example, a limitation may stop a user from entering a negative value into a field that only accepts positive numbers.

CONSTRAINTS

- ▶ **The following can be guaranteed via constraints**
- ▶ **Data Consistency** – The consistency of data in a database can be upheld by using constraints.
- ▶ These constraints are able to ensure that the primary key value in one table is followed by the foreign key value in another table.

CONSTRAINTS

- ▶ **The following can be guaranteed via constraints**
- ▶ **Data integrity** – The accuracy and completeness of the data in a database are ensured by constraints.
- ▶ For example, a constraint can stop a user from putting a null value into a field that requires one

CONSTRAINTS

- ▶ Entity Integrity Constraints
- ▶ A database management system uses entity integrity constraints (EICs) to enforce rules that guarantee a table's primary key is unique and not null. The consistency and integrity of the data in a database are maintained by EICs, which are created to stop the formation of duplicate or incomplete entries.
- ▶ Each item in a table in a relational database is uniquely identified by one or more fields known as the primary key. EICs make a guarantee that every row's primary key value is distinct and not null.

CONSTRAINTS

- ▶ Take the "Employees" table, for instance, which has the columns "EmployeeID" and "Name."
- ▶ The table's primary key is the EmployeeID column.
- ▶ An EIC on this table would make sure that each row's unique EmployeeID value is there and that it is not null.
- ▶ If you try to insert an entry with a duplicate or null EmployeeID, the database management system will reject the insertion and produce an error.

CONSTRAINTS

- ▶ This guarantees that the information in the table is correct and consistent.
- ▶ EICs are a crucial component of database architecture and assist guarantee the accuracy and dependability of the data contained in a database.

MySQL

- ▶ SELECT - extracts data from a database
- ▶ UPDATE - updates data in a database
- ▶ DELETE - deletes data from a database
- ▶ INSERT INTO - inserts new data into a database
- ▶ CREATE DATABASE - creates a new database
- ▶ ALTER DATABASE - modifies a database
- ▶ CREATE TABLE - creates a new table
- ▶ ALTER TABLE - modifies a table
- ▶ DROP TABLE - deletes a table
- ▶ CREATE INDEX - creates an index (search key)
- ▶ DROP INDEX - deletes an index

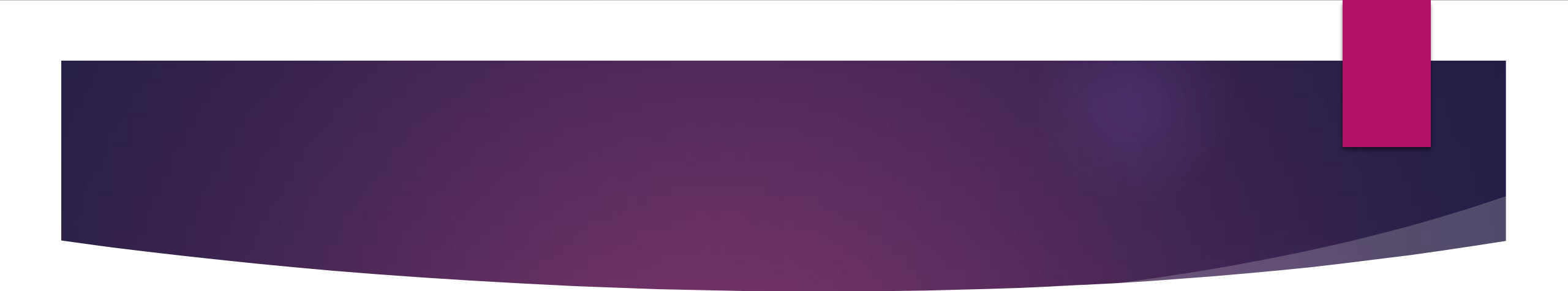
SQL

SQL

- ▶ SQL tutorial gives unique learning on **Structured Query Language** and it helps to make practice on SQL commands which provides immediate results.
- ▶ SQL is a language of database, it includes database creation, deletion, fetching rows and modifying rows etc.
- ▶ SQL is an ANSI (American National Standards Institute) standard, but there are many different versions of the SQL language.

What is SQL?

- ▶ SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in relational database.
- ▶ SQL is the standard language for Relation Database System.
- ▶ All relational database management systems like MySQL, MS Access, Oracle, Sybase, Informix, postgres and SQL Server use SQL as standard database language.

- 
- ▶ Also, they are using different dialects, such as:
 - ▶ ☐ MS SQL Server using T-SQL,
 - ▶ ☐ Oracle using PL/SQL,
 - ▶ ☐ MS Access version of SQL is called JET SQL (native format) etc.

Why SQL?

- ▶ Allows users to access data in relational database management systems.
- ▶ Allows users to describe the data.
- ▶ Allows users to define the data in database and manipulate that data.
- ▶ Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- ▶ Allows users to create and drop databases and tables.
- ▶ Allows users to create view, stored procedure, functions in a database.
- ▶ Allows users to set permissions on tables, procedures and views

History:

- ▶ **1970** -- Dr. E. F. "Ted" of IBM is known as the father of relational databases. He described a relational model for databases.
- ▶ **1974** -- Structured Query Language appeared.
- ▶ **1978** -- IBM worked to develop Codd's ideas and released a product named System/R.
- ▶ **1986** -- IBM developed the first prototype of relational database and standardized by ANSI. The first relational database was released by Relational Software and its later becoming Oracle.

SQL Process:

- ▶ When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.
- ▶ There are various components included in the process.
- ▶ These components are Query Dispatcher, Optimization Engines, Classic Query Engine and SQL Query Engine, etc.
- ▶ Classic query engine handles all non-SQL queries, but SQL query engine won't handle logical files.

SQL Commands

COMMANDS

Commands

- ▶ The standard SQL commands to interact with relational databases are CREATE, SELECT, INSERT, UPDATE, DELETE and DROP.
- ▶ These commands can be classified into groups based on their nature:

Nature of Commands

DDL – Data Definition Language

DML – Data Manipulation Language

DCL – Data Control Language

DQL – Data Query Language

Data Definition Language:

CREATE

- Creates a new table, a view of a table, or other object in database

ALTER

- Modifies an existing database object, such as a table.

DROP

- Deletes an entire table, a view of a table or other object in the database.

Data Manipulation Language

INSERT

- Creates records

UPDATE

- Modifies records

DELETE

- Delete records

Data Control Language

GRANT

- Gives a privilege to user

REVOKE

- Takes back privileges granted from user

Data Query Language

SELECT

- Retrieves certain records from one or more tables

CREATING A DATABASE

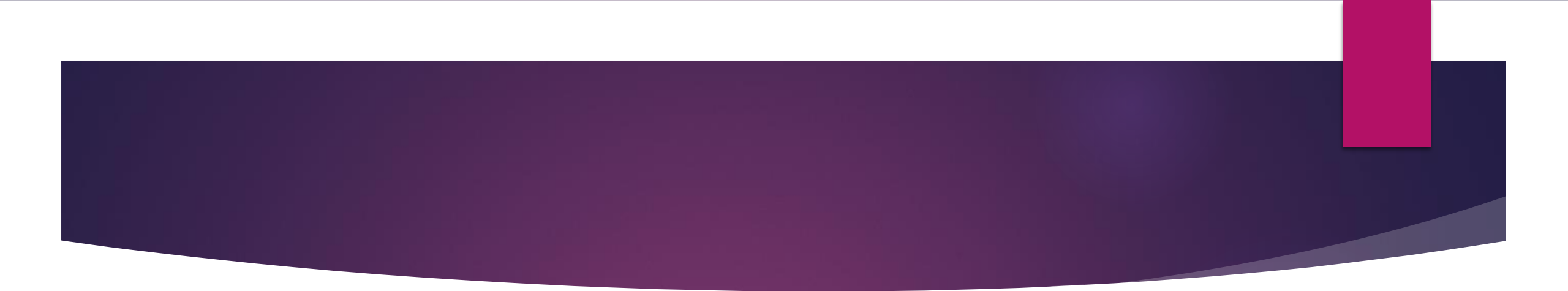
```
▶ CREATE TABLE CUSTOMERS (  
▶ ID INT NOT NULL,  
▶ NAME VARCHAR (20) NOT NULL,  
▶ AGE INT NOT NULL,  
▶ ADDRESS CHAR (25) ,  
▶ SALARY DECIMAL (18, 2) ,  
▶ PRIMARY KEY (ID)  
▶ );
```

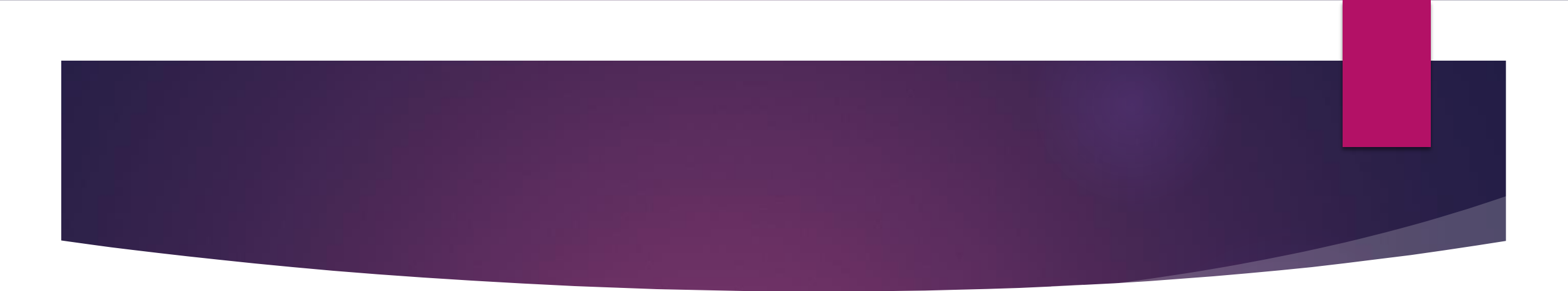
MODIFY

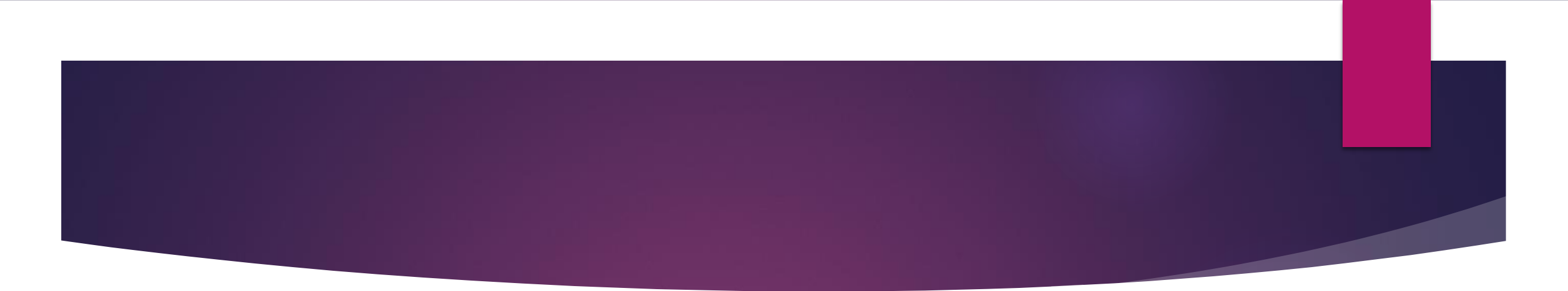
- ▶ If CUSTOMERS table has already been created, then to add a NOT NULL constraint to SALARY column in Oracle and MySQL, you would write a statement similar to the following:
- ▶ `ALTER TABLE CUSTOMERS`
- ▶ `MODIFY SALARY DECIMAL (18, 2) NOT NULL;`

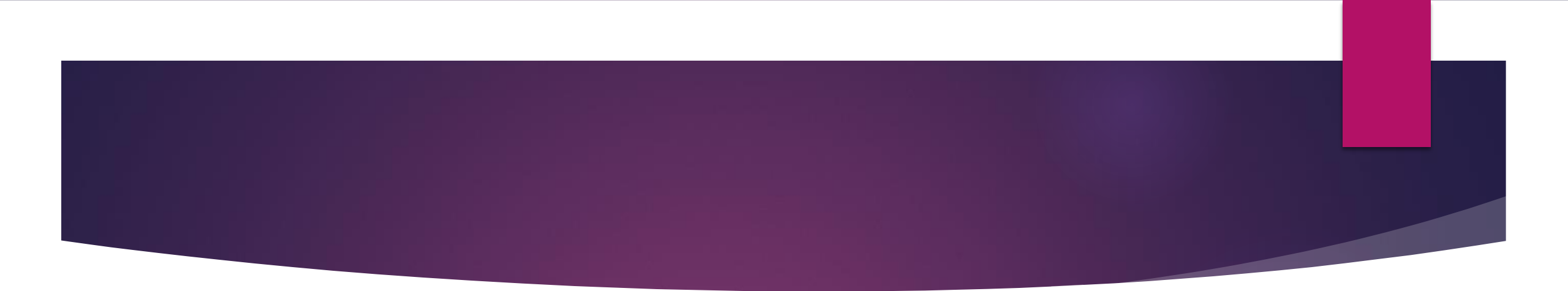
Database Users

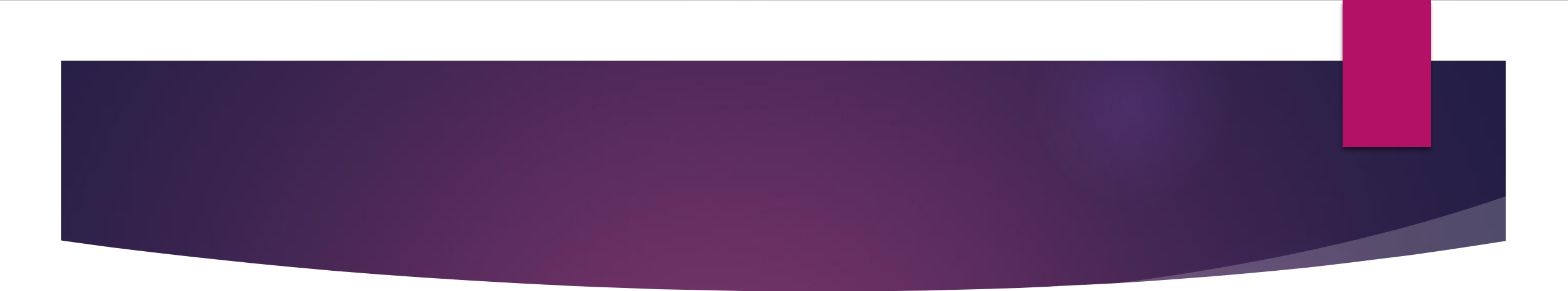
- ▶ In a database system, users are assigned specific permissions and roles to control their access and actions within the database.
- ▶ The permissions and roles determine what operations a user can perform and what data they can access.
- ▶ Here are the common types of permissions and roles in a database:

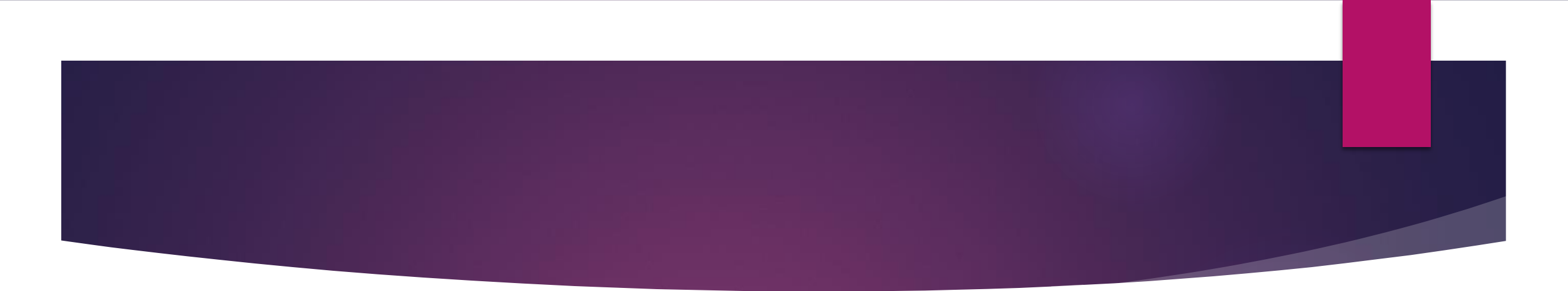
- 
- ▶ System Administrator (DBA): The system administrator, often referred to as the Database Administrator (DBA), has the highest level of access and privileges in the database system.
 - ▶ They have the authority to manage the overall database environment, including creating and configuring databases, managing user accounts, setting security policies, and performing backups and recoveries.

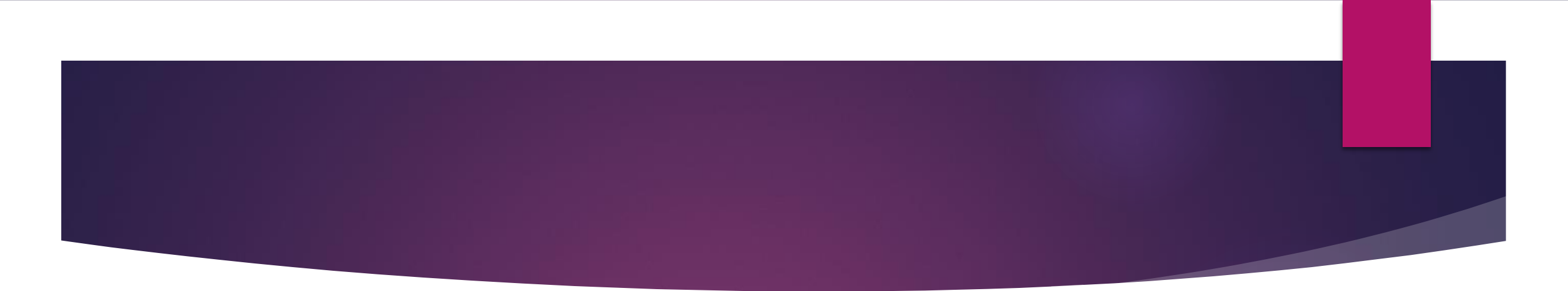
- 
- ▶ Database Owner: The database owner is a privileged user who has full control over a specific database.
 - ▶ They can manage database objects, define access permissions for other users, and perform administrative tasks within that database.
 - ▶ The database owner is typically responsible for maintaining the integrity and security of the database.

- 
- ▶ Data Administrator: Data administrators have permissions to manage the structure and organization of the database.
 - ▶ They can create and modify database tables, define relationships between tables, and establish data integrity constraints.
 - ▶ Data administrators also handle tasks such as database optimization, performance tuning, and data backup and recovery.

- 
- ▶ Data Entry/User: Users with data entry or user roles have limited permissions within the database. They can insert, update, and delete data records but usually do not have privileges to modify the database schema or perform administrative tasks. Data entry users primarily interact with the database to enter and retrieve data.
 - ▶ Read-Only Users: Read-only users have restricted permissions and can only view the data in the database. They cannot modify or delete records, tables, or other database objects. Read-only access is typically granted to users who need to analyze or retrieve information from the database without making any changes.

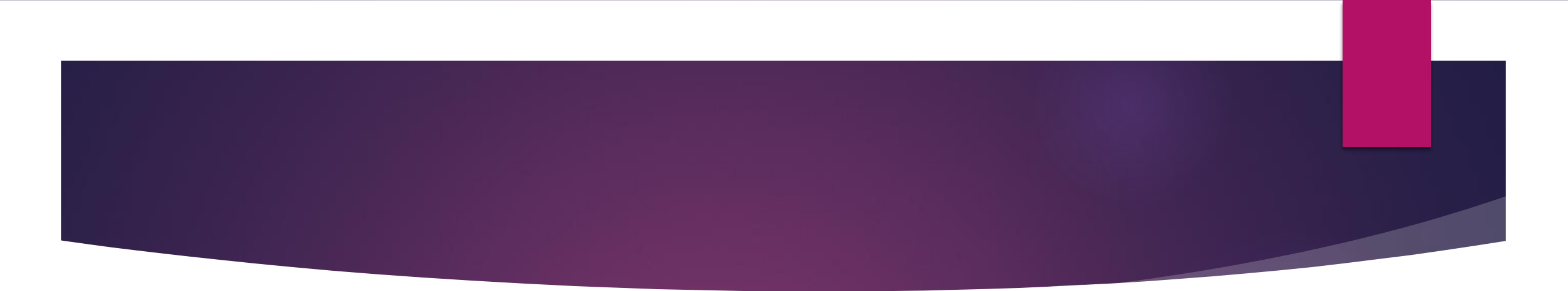
- 
- ▶ Application-Specific Roles: In some cases, users may have roles specific to certain applications or modules within the database. These roles define access and functionality based on the requirements of the application. For example, an application may have roles for managers, supervisors, and regular employees, each with different levels of access and permissions.

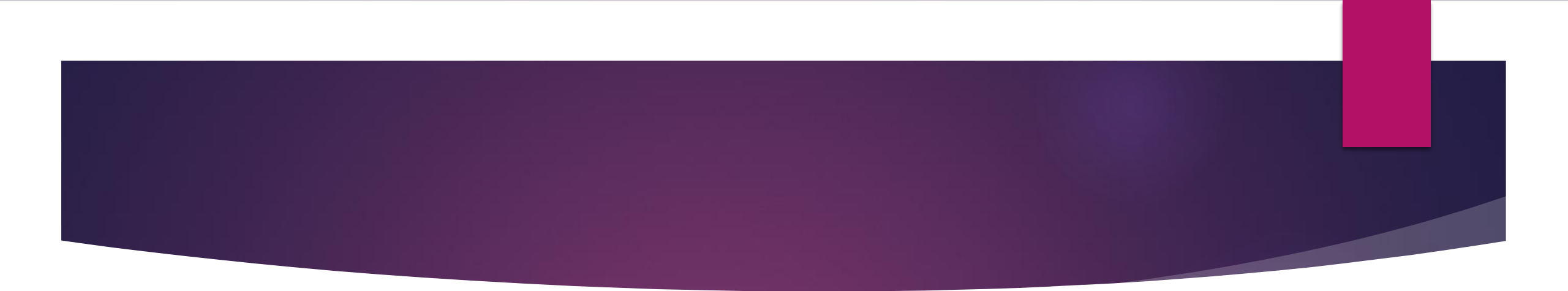
- 
- ▶ Custom Roles: Database systems often provide the ability to create custom roles with specific sets of permissions.
 - ▶ Custom roles allow for more fine-grained control over user access.
 - ▶ Organizations can define roles based on their specific requirements and assign appropriate permissions to each role

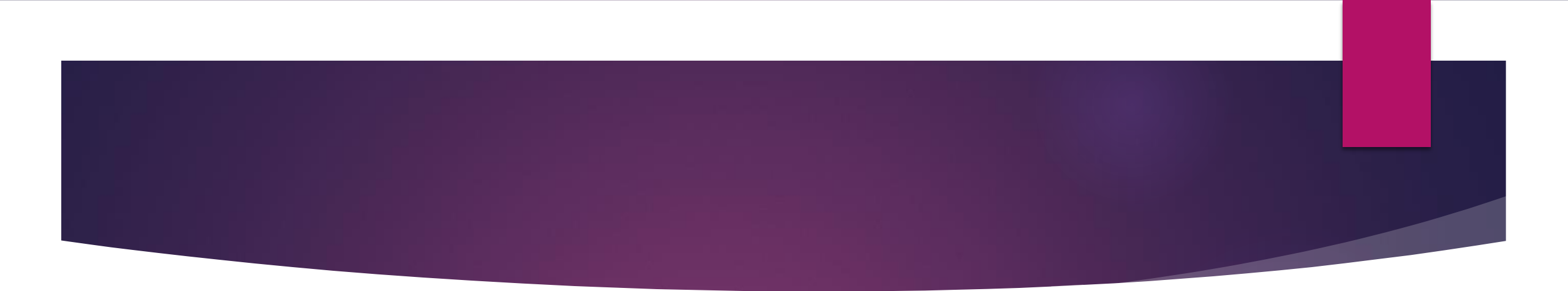
- 
- ▶ Permissions and roles can be managed through the database management system's security mechanisms.
 - ▶ These mechanisms typically include user account management, access control lists, and privileges assignment.
 - ▶ It's important to carefully assign permissions and roles to ensure data security, integrity, and appropriate access levels for different users within the database system

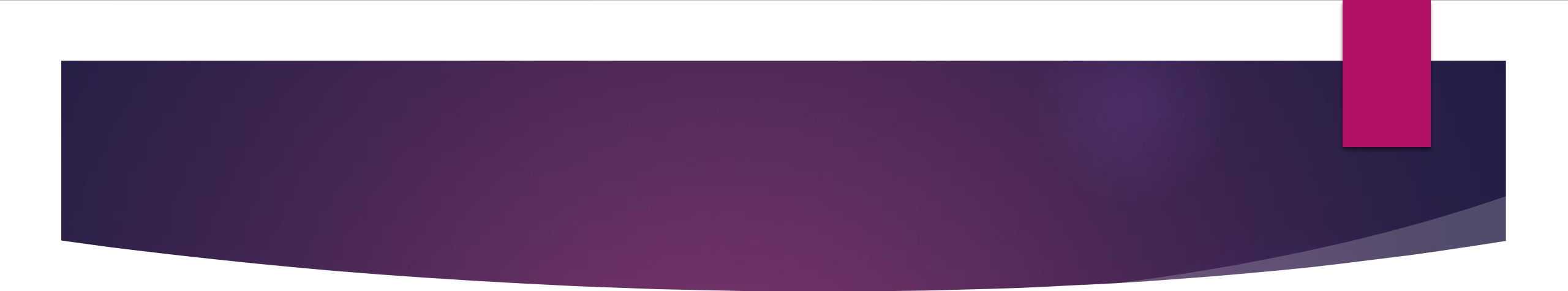
Data Connections

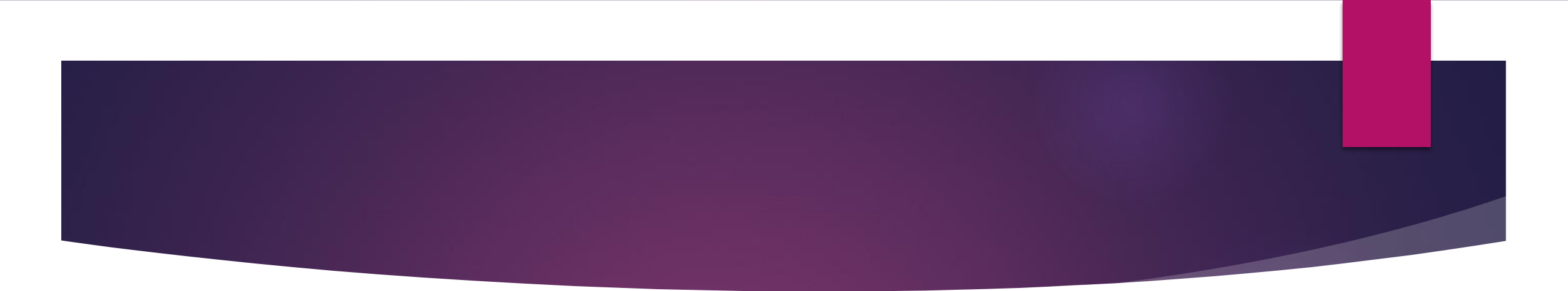
- ▶ To establish data connections in a database, various techniques and protocols are used. Here are some commonly employed techniques:
- ▶ TCP/IP Protocol: Transmission Control Protocol/Internet Protocol (TCP/IP) is the fundamental protocol used for establishing data connections over the internet.
- ▶ TCP/IP ensures reliable and secure data transmission between client and server applications. It enables the establishment of connections using IP addresses and port numbers.

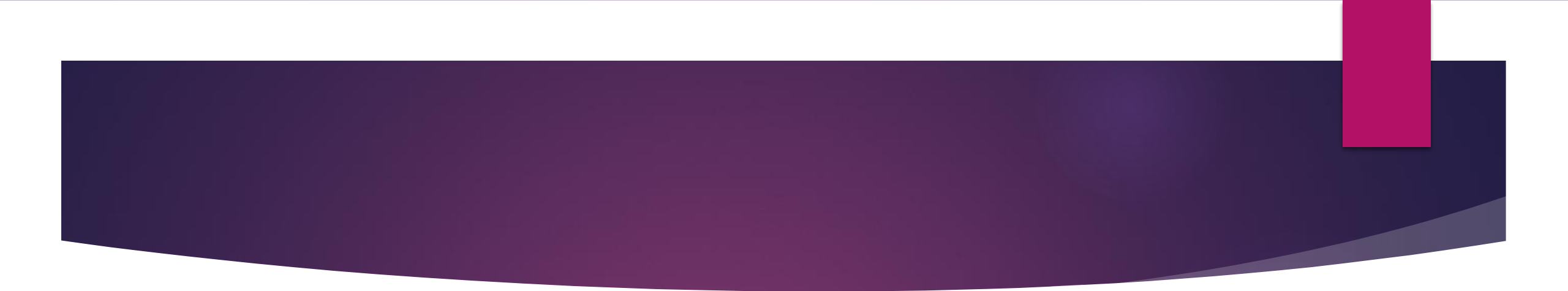
- 
- ▶ ODBC (Open Database Connectivity): ODBC is a standardized database API (Application Programming Interface) that allows applications to access and interact with different database systems.
 - ▶ ODBC provides a driver manager and database drivers specific to each database system, enabling applications to establish connections using a common interface.

- 
- ▶ JDBC (Java Database Connectivity): JDBC is a Java API that provides a set of classes and interfaces for connecting Java applications to various databases.
 - ▶ It allows Java applications to establish connections using JDBC drivers specific to the targeted database system.
 - ▶ JDBC provides a platform-independent way to interact with databases.

- 
- ▶ ORM (Object-Relational Mapping) Frameworks: ORM frameworks such as Hibernate, Entity Framework, or Django ORM provide abstraction layers between the application code and the database.
 - ▶ These frameworks use configuration or annotations to establish database connections and map objects to database tables.
 - ▶ ORM frameworks often simplify the process of connecting to databases and performing database operations.

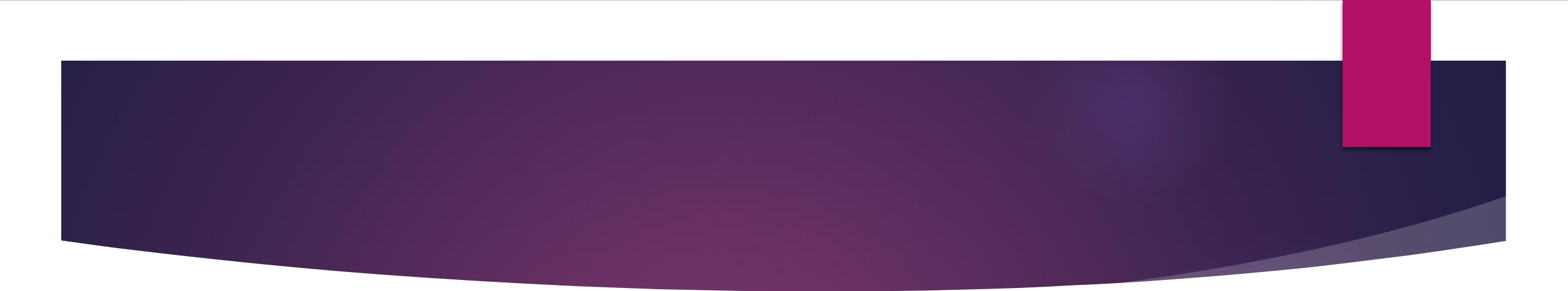
- 
- ▶ Connection Strings: Connection strings are typically used in database connection settings to specify the necessary information for establishing a connection.
 - ▶ A connection string includes parameters such as the server or host name, port number, database name, authentication credentials, and other relevant connection details.
 - ▶ Applications use these connection strings to establish connections programmatically.

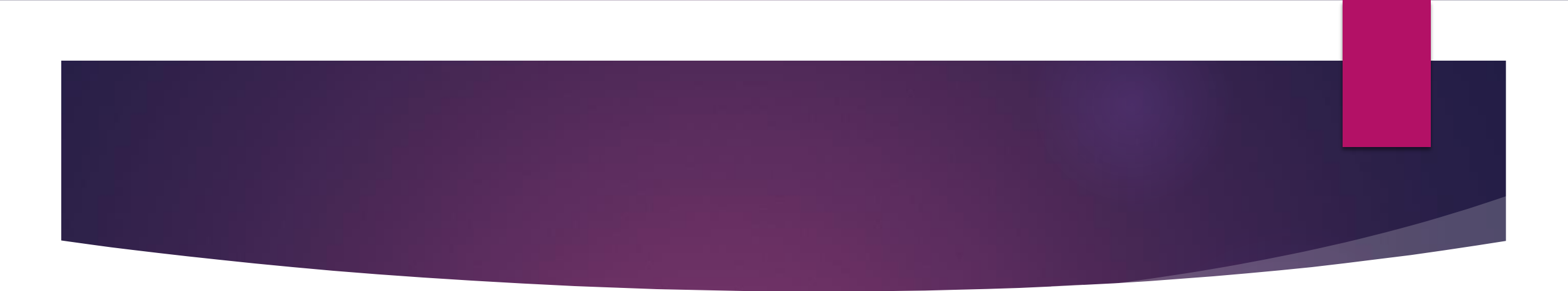
- 
- ▶ Connection Pooling: Connection pooling is a technique that involves creating and maintaining a pool of pre-established database connections.
 - ▶ Instead of creating a new connection for every user request, connection pooling reuses existing connections from the pool, reducing the overhead of establishing new connections.
 - ▶ This technique improves performance and scalability in applications with frequent database interactions.

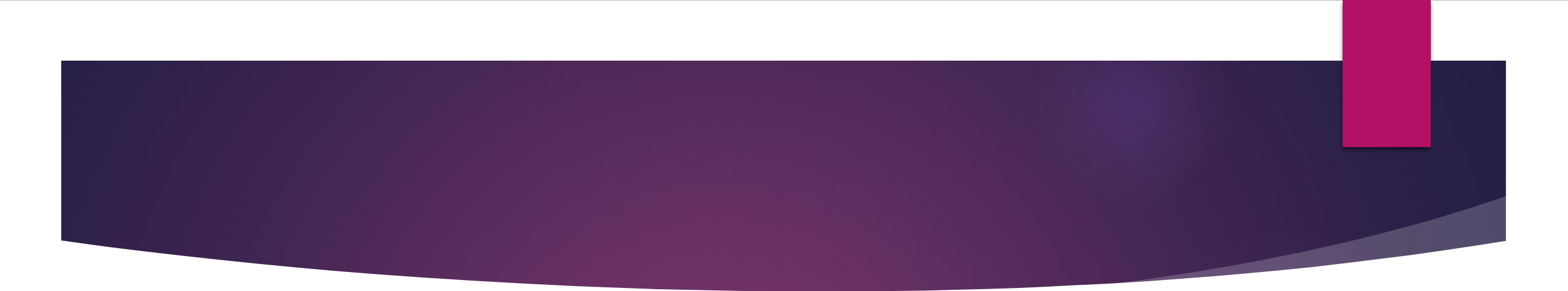
- 
- ▶ Web Service APIs: Some databases provide web service APIs that allow applications to establish data connections and interact with the database using web protocols such as HTTP. These APIs often expose methods or endpoints for querying, modifying, and retrieving data from the database.
 - ▶ These techniques facilitate the establishment of data connections between applications and databases, enabling efficient and secure data retrieval, manipulation, and storage. The choice of technique depends on the programming language, database system, and specific requirements of the application.

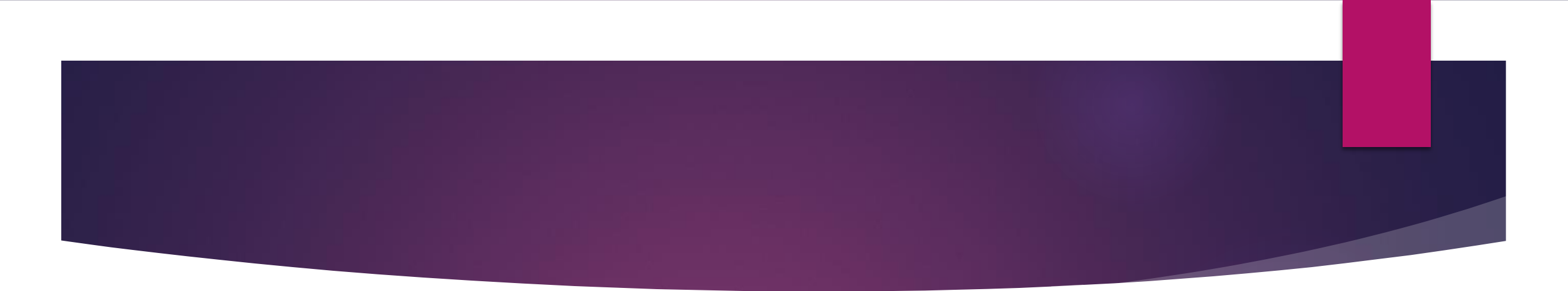
Transactions Concept in Database

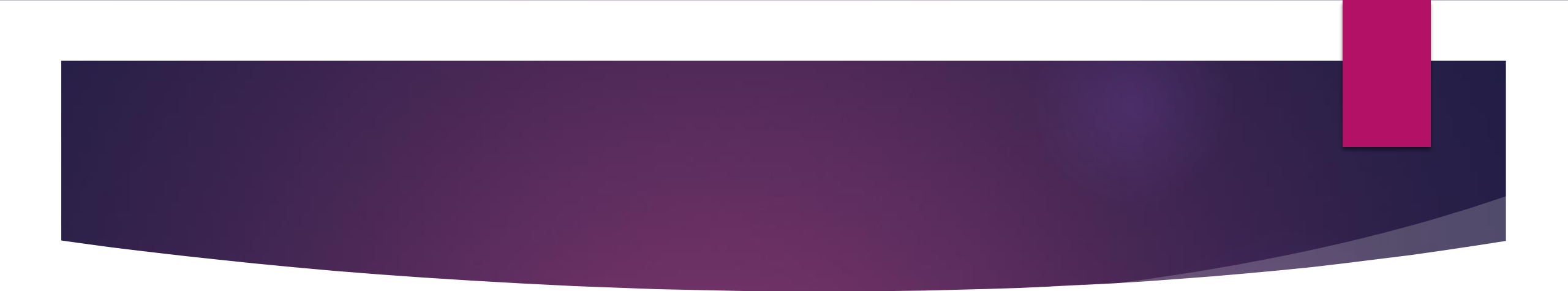
TRANSACTIONS

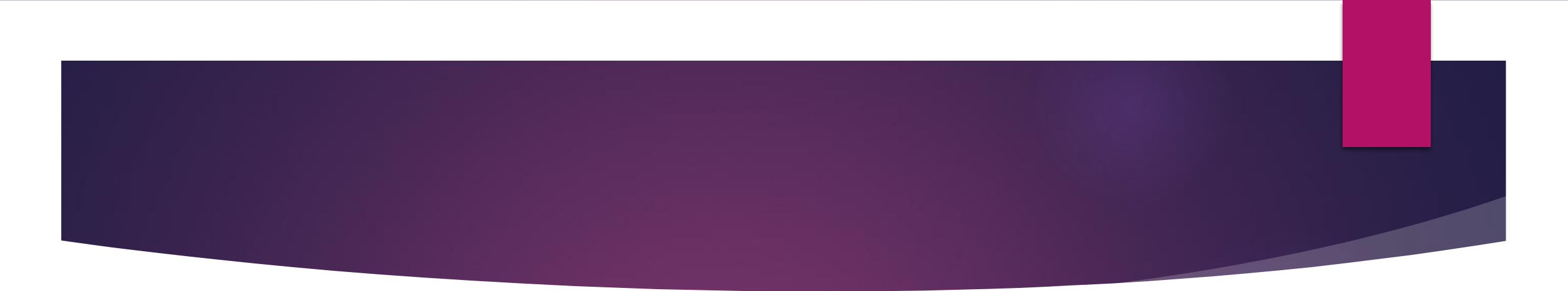
- 
- ▶ The transaction concept in a database refers to a logical unit of work that consists of multiple database operations.
 - ▶ A transaction groups a set of operations together to ensure that they are executed atomically, consistently, and reliably.
 - ▶ The transaction concept is crucial for maintaining data integrity and concurrency control in a database system.
 - ▶ Here are the key characteristics and properties of a transaction:

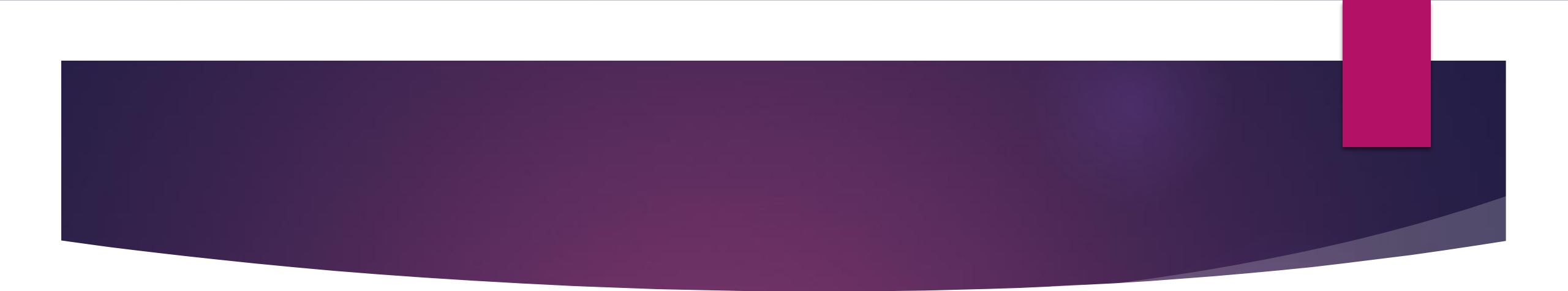
- 
- ▶ Atomicity: Atomicity ensures that a transaction is treated as a single, indivisible unit of work.
 - ▶ It means that either all the operations within the transaction are completed successfully, or none of them take effect.
 - ▶ If any operation fails or encounters an error, the entire transaction is rolled back, and the database is restored to its original state before the transaction started.

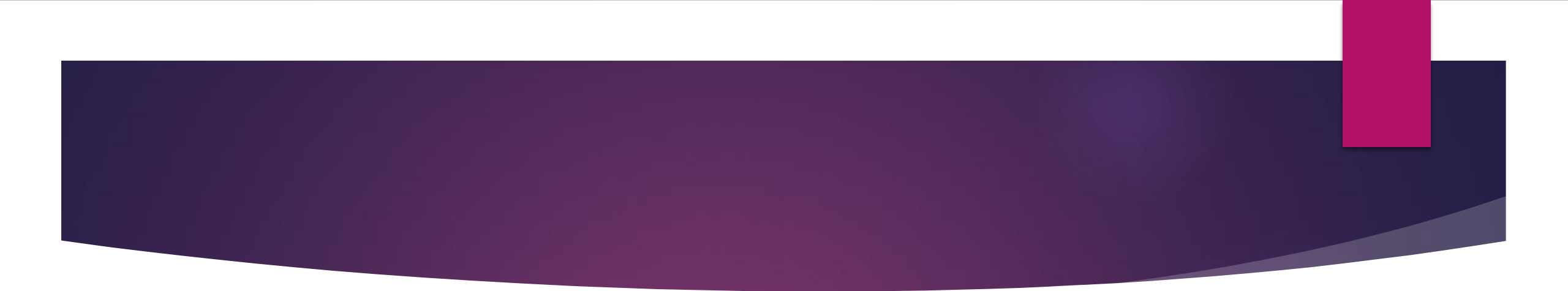
- 
- ▶ Consistency: Consistency ensures that a transaction brings the database from one consistent state to another consistent state.
 - ▶ The database must satisfy predefined integrity constraints and rules before and after the transaction.
 - ▶ If a transaction violates any integrity constraints, it is rolled back, and the database remains unchanged

- 
- ▶ Isolation: Isolation ensures that each transaction is executed in isolation from other concurrent transactions.
 - ▶ Even if multiple transactions are running simultaneously, each transaction should perceive the database as if it is executing in isolation.
 - ▶ This prevents interference and ensures that concurrent transactions do not conflict or affect each other's intermediate results.

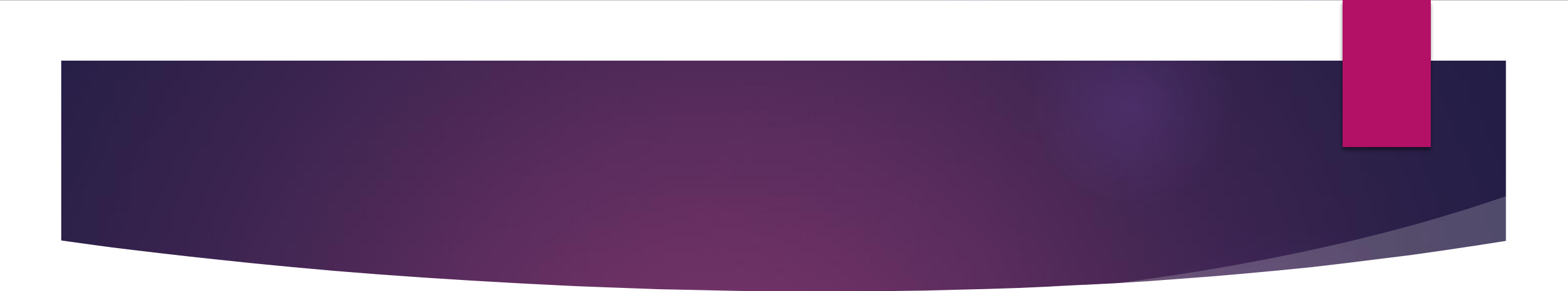
- 
- ▶ Durability: Durability guarantees that once a transaction is committed and its changes are written to the database, they persist and are not lost, even in the event of system failures or crashes.
 - ▶ Committed data is stored in non-volatile storage, such as hard disks or SSDs, to ensure its durability and availability for future transactions.

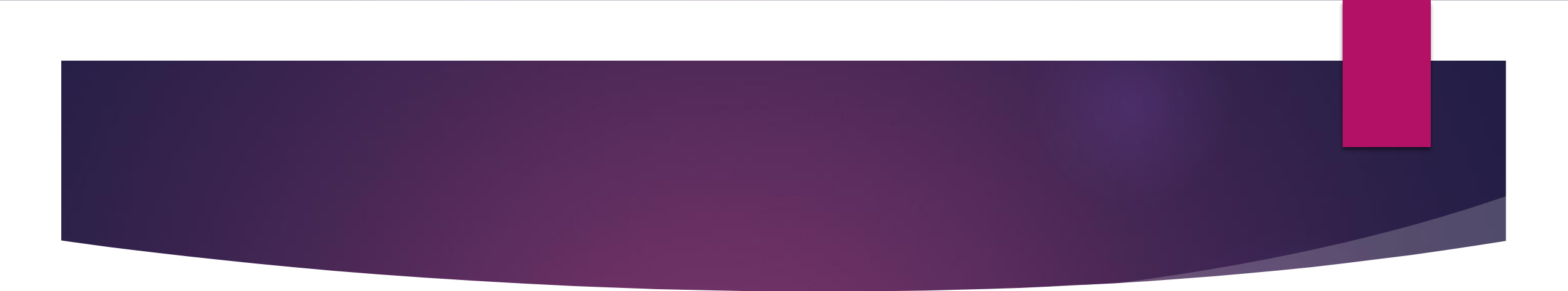
- 
- ▶ The ACID properties (Atomicity, Consistency, Isolation, and Durability) collectively describe the ideal behavior and guarantees provided by a transaction in a database system.
 - ▶ These properties ensure that transactions are reliable, robust, and maintain data integrity in the face of concurrent access, system failures, or unexpected errors.

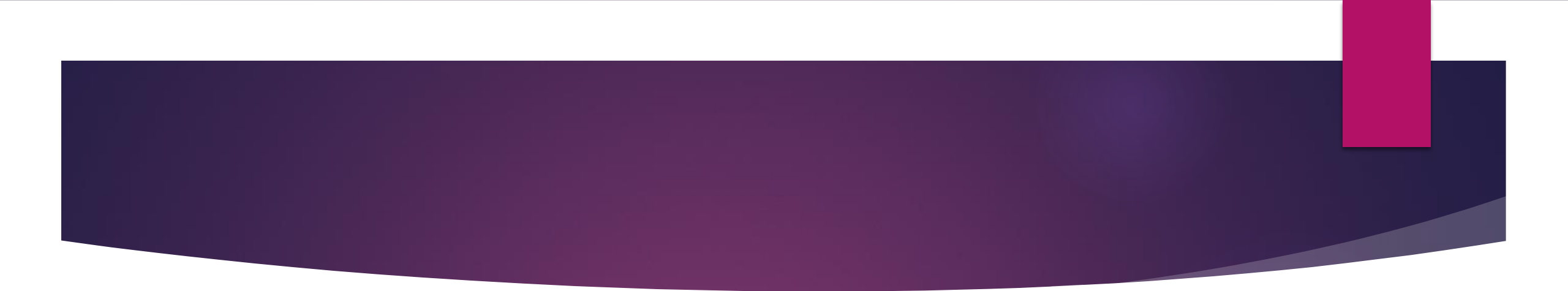
- 
- ▶ Transaction management is typically handled by the database management system (DBMS) through mechanisms such as transaction logs, locking, and concurrency control protocols.
 - ▶ Developers can use programming interfaces and query languages to define and manage transactions explicitly within their applications.

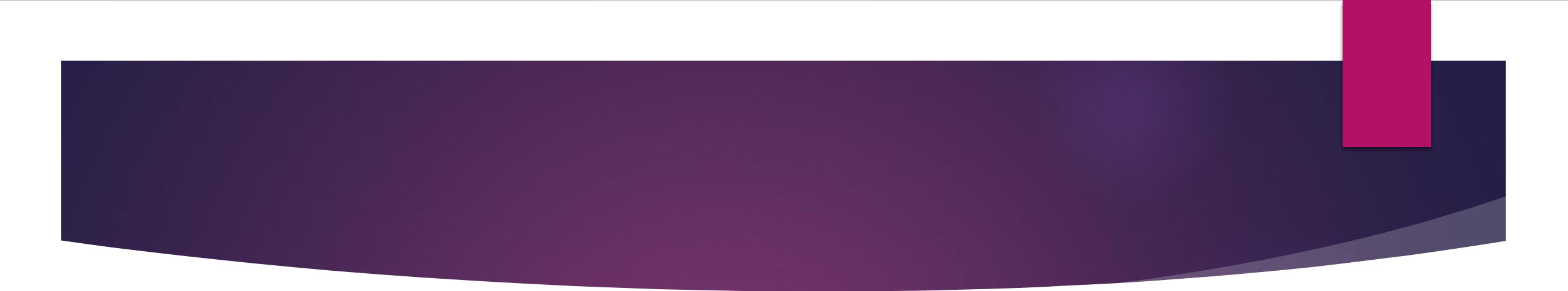
- 
- ▶ By utilizing the transaction concept, applications can ensure data integrity, recoverability, and concurrency control in multi-user environments where multiple transactions may be executing simultaneously.
 - ▶ Transactions allow complex operations and modifications to the database to be carried out reliably and consistently, making them a fundamental concept in database systems

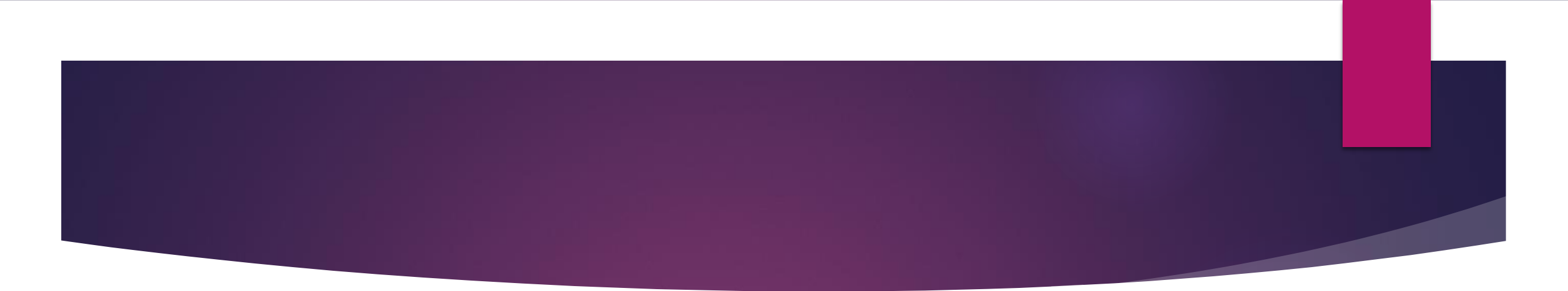
Real Life Examples of Transactions

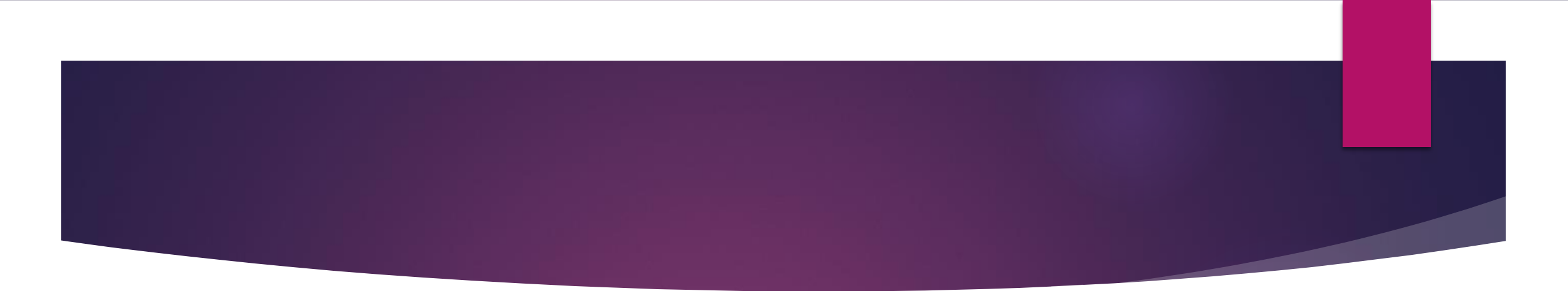
- 
- ▶ Here are some real-life examples of transactions in a database:
 - ▶ Online Purchase: When a customer makes an online purchase, a transaction is initiated.
 - ▶ The transaction may involve multiple operations such as deducting the purchase amount from the customer's account, updating inventory levels, generating an invoice, and recording the transaction details.
 - ▶ If any operation fails during the transaction, the entire purchase is rolled back to maintain data consistency.

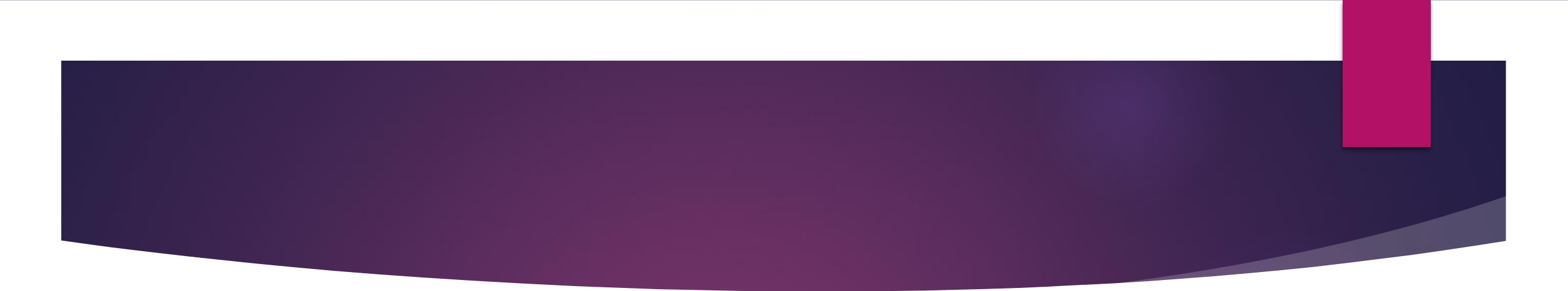
- 
- ▶ Fund Transfer: When a customer transfers funds from one bank account to another, a transaction is executed.
 - ▶ The transaction involves debiting the sender's account and crediting the recipient's account.
 - ▶ These operations are performed atomically to ensure that the money is either transferred successfully or not at all.
 - ▶ In case of any failure, the transaction is rolled back to maintain data integrity.

- 
- ▶ Airline Reservation: When a customer books a flight ticket, a transaction takes place.
 - ▶ The transaction includes reserving a seat on the flight, deducting the ticket fare from the customer's account, updating seat availability, and storing the booking details.
 - ▶ If any operation fails during the transaction, the reservation is canceled, and the database is rolled back to its previous state.

- 
- ▶ ATM Withdrawal: When a customer withdraws cash from an ATM, a transaction occurs.
 - ▶ The transaction involves debiting the customer's account, updating the account balance, and recording the withdrawal details.
 - ▶ If any operation fails during the transaction, the withdrawal is canceled, and the account balance remains unchanged.

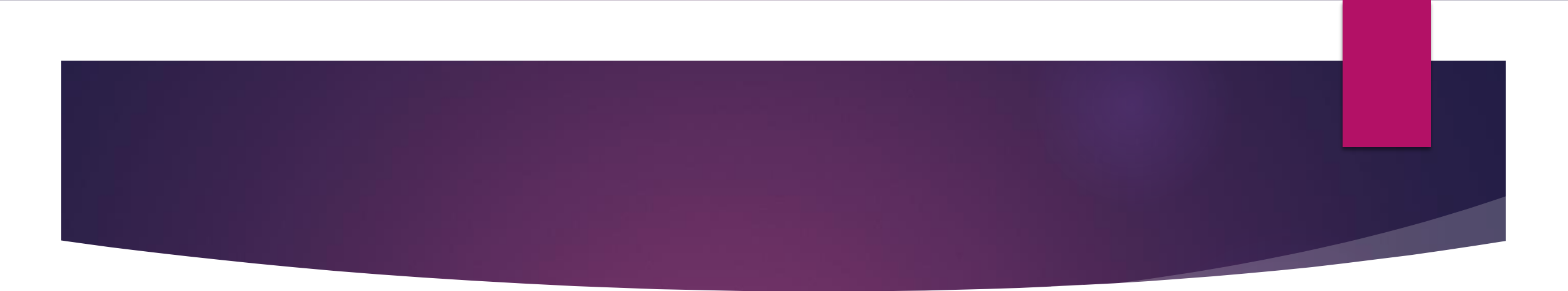
- 
- ▶ Hotel Reservation: When a customer makes a hotel reservation, a transaction is initiated.
 - ▶ The transaction involves reserving a room, updating room availability, recording guest details, and charging the reservation amount.
 - ▶ If any operation fails, the reservation is canceled, and the database is rolled back to ensure consistency.

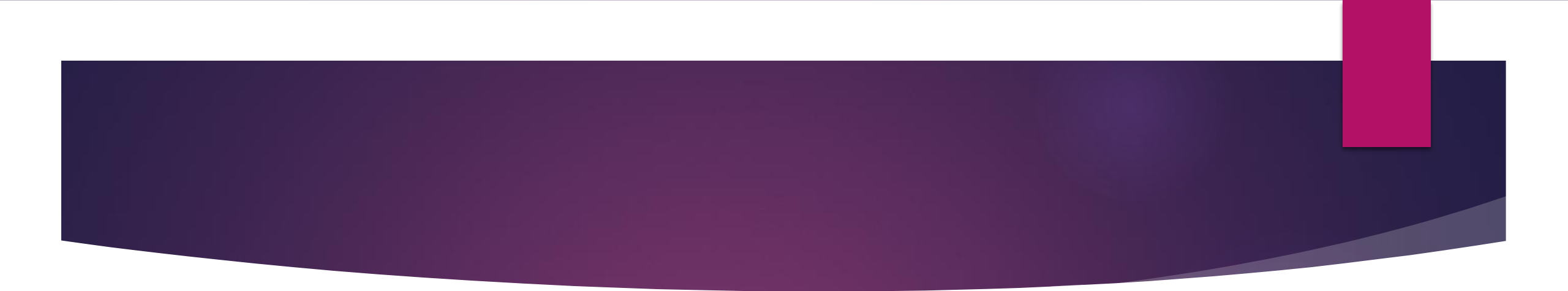
- 
- ▶ Stock Trading: When an investor places a stock trade order, a transaction is executed.
 - ▶ The transaction involves checking the availability of stocks, updating the investor's portfolio, executing the trade, and updating the stock market data.
 - ▶ If any operation fails or the trade cannot be executed, the transaction is rolled back, and the investor's portfolio remains unchanged.

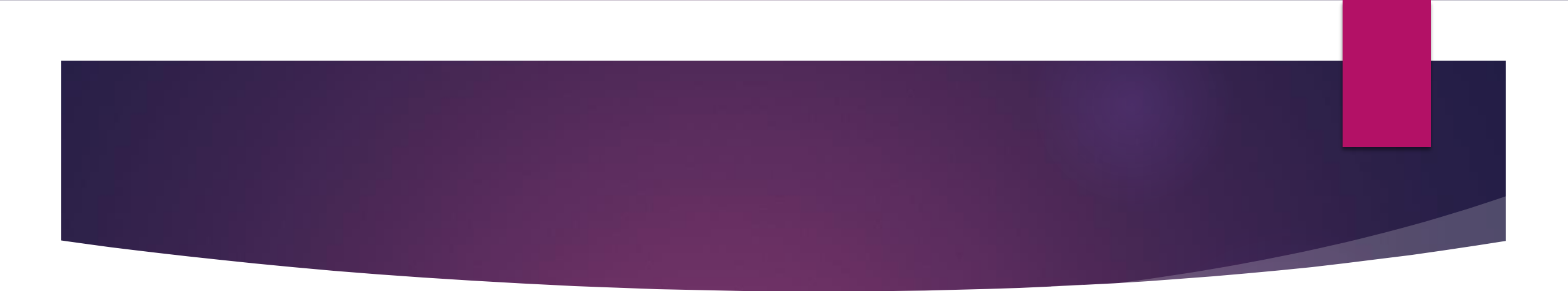
- 
- ▶ In these examples, transactions are used to ensure that multiple operations related to a specific task are executed reliably and consistently.
 - ▶ If any operation within the transaction fails, the entire transaction is rolled back to maintain data integrity and to ensure that the database remains in a consistent state.

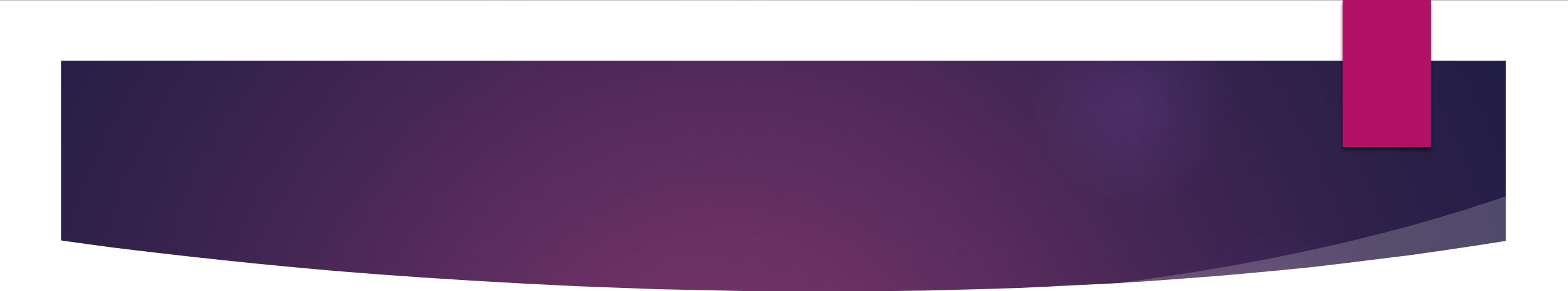
Serialization Concepts

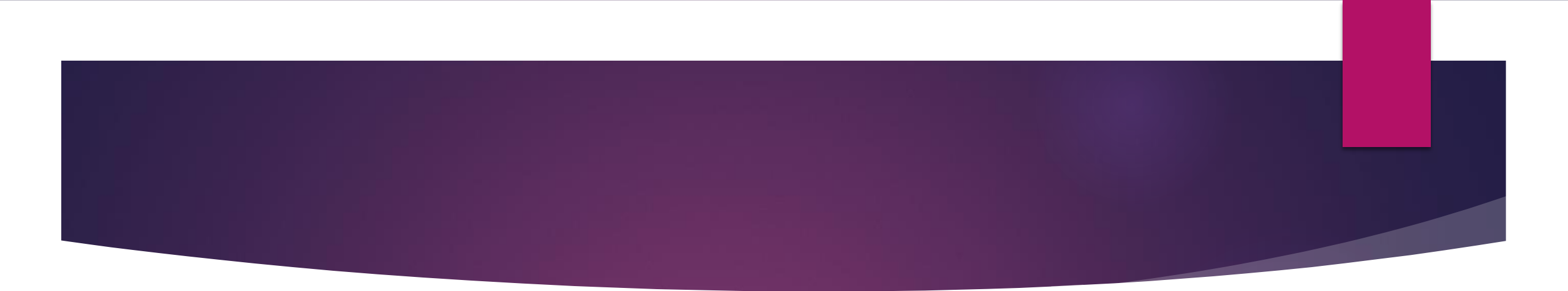
- ▶ In the context of database transactions, serialization refers to the concept of executing transactions in a serial or sequential order, as if they were executed one after another.
- ▶ It ensures that concurrent transactions appear to execute in a serialized manner, preserving the consistency of the database and avoiding conflicts or data anomalies.

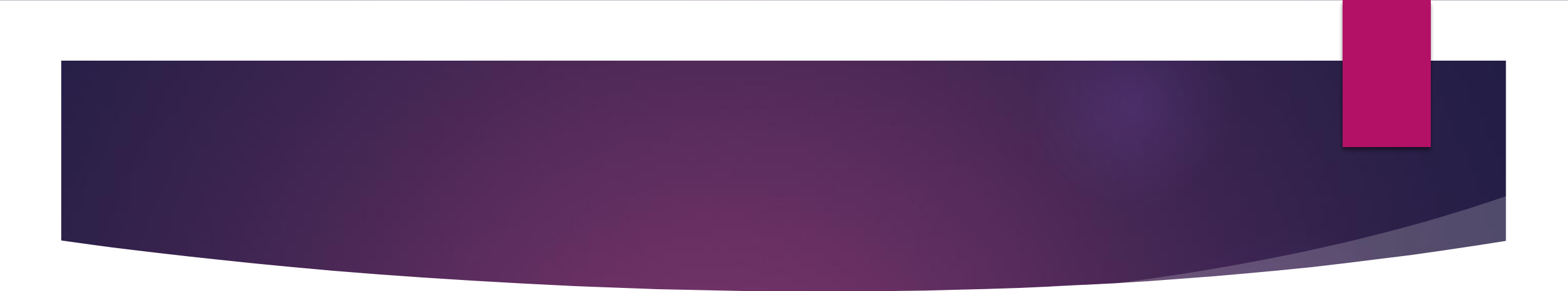
- 
- ▶ The primary goal of serialization is to maintain data integrity and consistency by preventing concurrent transactions from interfering with each other.
 - ▶ When transactions are executed concurrently, there is a potential for conflicts to arise, such as lost updates, dirty reads, or inconsistent results.
 - ▶ Serialization ensures that the final outcome of concurrent transactions is equivalent to a sequential execution of those transactions.

- 
- ▶ To achieve serialization, database systems employ concurrency control mechanisms.
 - ▶ These mechanisms manage access to shared resources (e.g., data objects, tables, indexes) and coordinate the execution of concurrent transactions. Two commonly used techniques for serialization are:

- 
- ▶ Lock-Based Concurrency Control: In this technique, transactions acquire locks on resources they need to access.
 - ▶ Locks can be exclusive (write locks) or shared (read locks).
 - ▶ A transaction must obtain the required locks before accessing a resource and release them after completing the operation.
 - ▶ This ensures that conflicting operations from different transactions are serialized and executed in a controlled manner.

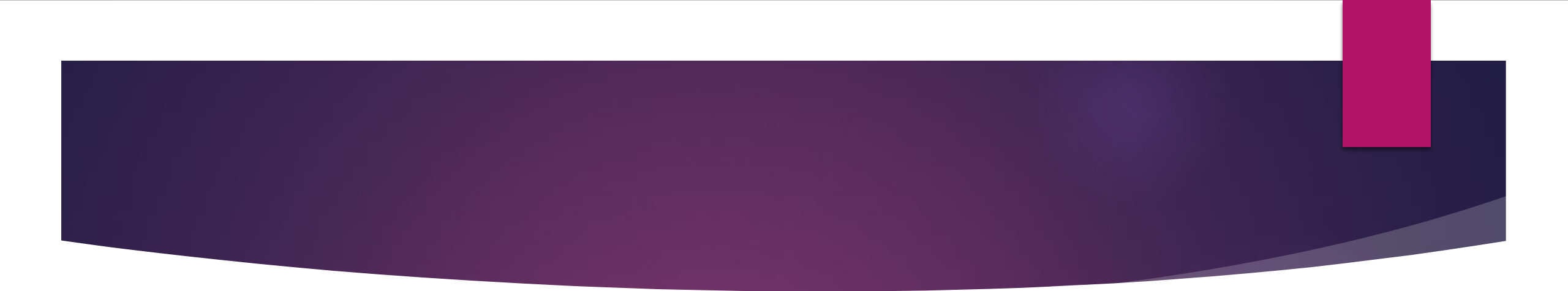
- 
- ▶ **Timestamp-Based Concurrency Control:** In this technique, each transaction is assigned a unique timestamp that represents its order of execution.
 - ▶ The system uses these timestamps to determine the serialization order of transactions.
 - ▶ Transactions with earlier timestamps are allowed to execute first, while conflicting operations by later transactions are delayed or rolled back to maintain serializability.

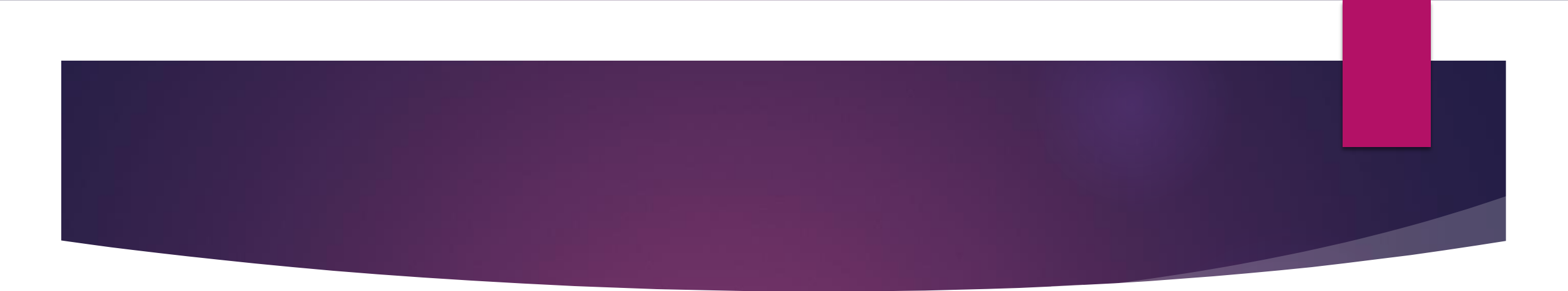
- 
- ▶ By enforcing serialization, database systems guarantee that concurrent transactions do not violate integrity constraints, maintain data consistency, and produce correct results.
 - ▶ However, serialization can limit concurrency and introduce performance overhead, as transactions may need to wait for locks or be delayed due to timestamp ordering.
 - ▶ Database systems employ various optimization techniques, such as deadlock detection and prevention, to minimize the impact on performance while ensuring data integrity.

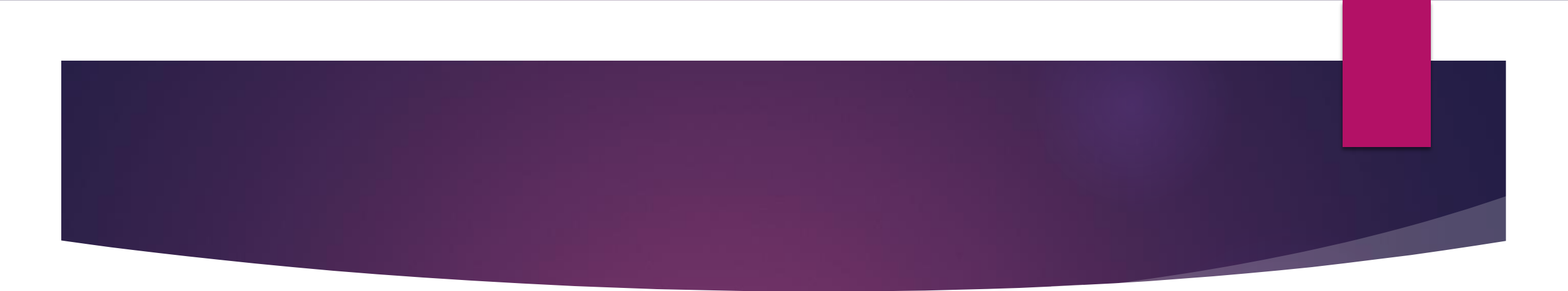
- 
- ▶ It's important to note that not all transactions need strict serialization.
 - ▶ Some transactions may not conflict with each other and can execute concurrently without compromising data consistency.
 - ▶ Database systems strive to strike a balance between concurrency and serialization to provide efficient and reliable transaction processing

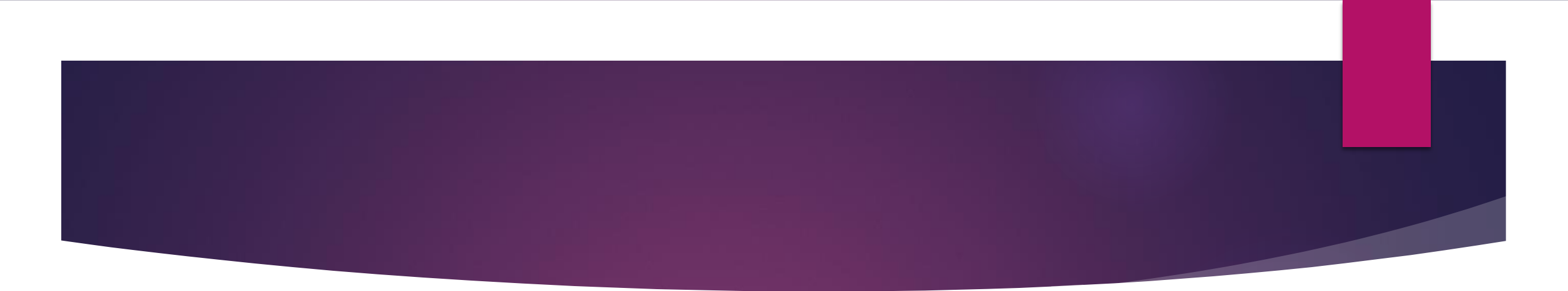
Transaction States

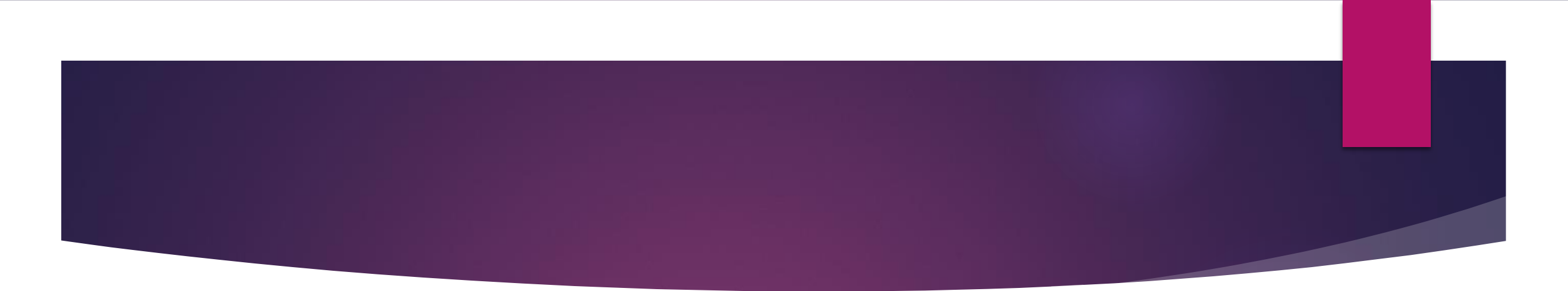
- ▶ In a database management system, transactions progress through different states as they are executed.
- ▶ These transaction states reflect the various stages of a transaction's lifecycle and determine its behavior and interaction with the database. The common transaction states are:

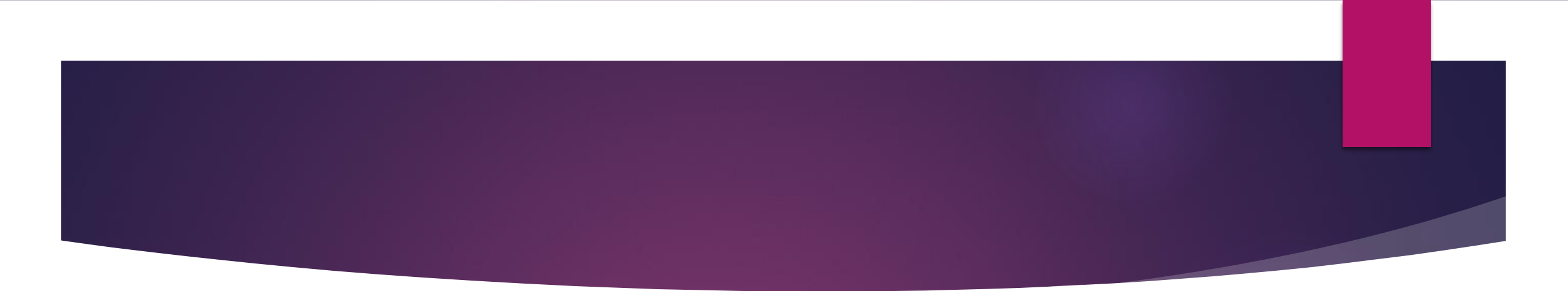
- 
- ▶ Active: The transaction begins in the active state when it starts its execution.
 - ▶ In this state, the transaction performs various database operations like reading or modifying data.

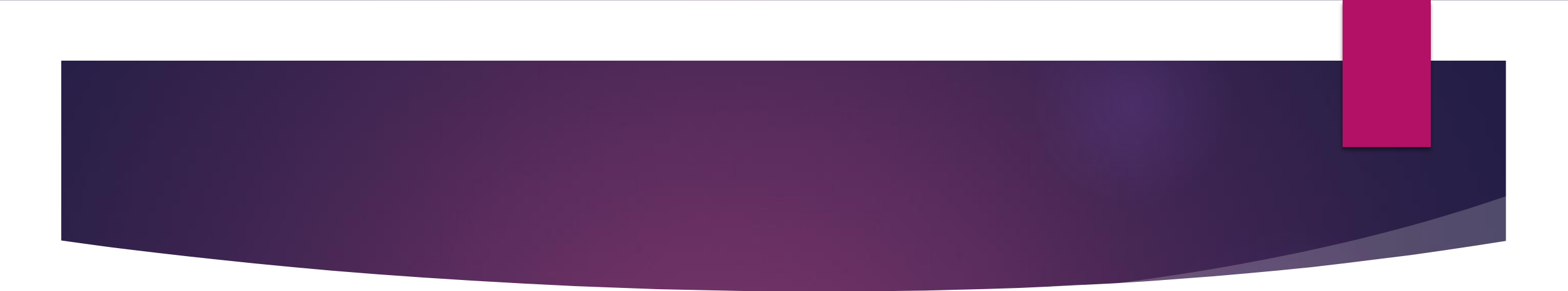
- 
- ▶ Partially Committed: After the transaction has executed all its operations successfully, it enters the partially committed state.
 - ▶ In this state, the transaction is yet to be permanently committed to the database.
 - ▶ However, the changes made by the transaction are temporarily visible to other transactions

- 
- ▶ Committed: Once a transaction is committed, it enters the committed state.
 - ▶ In this state, the transaction's changes are made permanent and durable in the database.
 - ▶ The committed state ensures that the changes are visible to other transactions, and they become a permanent part of the database.

- 
- ▶ Aborted: If a transaction encounters an error or failure during its execution and cannot proceed further, it is aborted. In the aborted state, all the changes made by the transaction are rolled back, and the database is restored to its state before the transaction started.
 - ▶ Aborting a transaction ensures that the database remains consistent despite any errors or failures.

- 
- ▶ Failed: The failed state occurs when a transaction encounters a critical error or system failure that prevents it from continuing its execution.
 - ▶ Unlike the aborted state, failed transactions are typically handled by the database system itself.
 - ▶ Recovery mechanisms are employed to handle failed transactions and restore the database to a consistent state.

- 
- ▶ Terminated: The terminated state represents the final state of a transaction.
 - ▶ It occurs when a transaction has either committed or aborted and has completed its execution.
 - ▶ In this state, the transaction is no longer active and does not interact with the database.

- 
- ▶ These transaction states are crucial for ensuring data integrity, consistency, and concurrency control in a database system.
 - ▶ They help maintain the correctness of database operations and allow multiple transactions to execute concurrently while preserving the integrity of the database.
 - ▶ Transaction management systems and database engines handle the transitions between these states and ensure the proper execution and handling of transactions