Oracle Academy

Notes

Only relevant pages and slides have been summarized.

Dedicated to the person I hate
more than anything or anyone
else in the entire world,

Myself.

Slide 1_2

Difference data and information
Data: collected facts about a topic or item
Information: the result of combining, comparing and performing data

As an example, data can be that in 2015 the company made 1,000,000 USD and in 2016 2,000,000 USD whereas information would be derived from that, like next years budget.

Other more complex examples can include the data from number of graduates, the % of students that passed a certain exam en the test scores. Information that can be derived from that could be the number of students that passed an exam by more than X percent. If you add class attendance you can calculate information such as “on average students that went to more than 80% of classes scored x% higher than other students.

Definition of Database
Centralized and structured set of data stored on a computer that provides the ability to add, modify and delete data. The data can be used to derive information from.

Introduction to relational databases
A relational database stores information *in* tables *with* rows and columns.

A table is a collection of records. A row is called a record (or instance). A column is referred to as a field (or attribute)

What makes a relational database relational is the fact that different tables can be linked to other tables through keys. For example each customer has an ID. That way when you have to enter in customer details you don’t need to enter data that exists elsewhere. So you just refer to the customer ID.

DBMS - database management system
A DMBS is software that controls the storage, organization and retrieval of data

Slide 1_3

Database development process

Conceptual data modeling > Database design > Database build

Different models
Relational, object oriented, flat file, hierarchical and network models.

Flat File Model
Designed around a single table, plain text, each line holds 1 record, fields are separated with delimiters such as tabs and commas.

Hierarchical Model
Organized like a tree structure, data is stored as records that are connected to one another through links, a record are a collection of field. Record is to a hierarchical model who a row is to a relational one.

Network Model
Can be regarded as a flexible way of representing objects and their relationships. Its comprised of a collection of records connected to one another through links (boxes = fields, lines = links)
Each record is a collection of fields, each of which contains only one data value. A link is an association between two records.

Object Oriented
An entity is modeled as an object, every object has a state (the set of values for the attributes of the object) and a behavior (set of methods the operate on the state of the object) the relationship between the objects is through sharing of access, An object must belong only one class as an instance of that class. You can derive a new class (subclass) from an existing class (superclass.)

Relational
Data is represented as a collection of tables, each column represents attributes that belong to the table. Each row represent an instance of the table. Each table is the visual representation of columns and rows. Every table has a field or a set of fields that uniquely identifies the row.

The order of the rows and columns is not important. Every row is unique, each field can contain only one value. Values within a column or field are from the slim domain (data type), tables must be unique and column names within each table must be unique.

Slides 1_4

Why do you need a database?
Integration of multiple components, multiple data items, multiple users.

Importance of business rules
It's important to identify and document business rules when designing a database.

Business rules: Allow the developer/architect to understand the relationship and constraints of the participating entities, help you understand the standardization procedure that an organization follows when handling huge data, should be simple and easy to understand and it must be kept up to date.

Business rules are used to understand the business processes and the nature, role, and scope of the data. Business rules help you categorize and design database tales, and are usually provided by managers, policy makers, documentation and operation manuals, organizational procedures and standards and interviews with end users.

Business rules and conceptual modeling
A conceptual model is important because

- describes the needs of the business
- Facilitates discussion
- Prevents mistakes and misunderstandings
- Forms important "ideal system" documentation
- Forms the basis of the design
- Documents the process of the business (which is the business rule)
- Takes into account regulations and laws governing this industry

Not all business rules can be modeled in a database

Slides 2_1

Single table
A flat file database is a type of database that stores data in a single table. They are generally plain-text form where *each line holds only record*.

Disadvantages and advantages of a flat file database

Advantage	Disadvantages
Easy to understand	Less security
Easy to inplement	Data inconsistency
Easy to extract information	Data redundancy
All records stored in one place	Cumbersome sharing of information
Simple sorting and filtering of reports	Slow for huge databases
Less hardware and software requirements.	

Relational databases

Presents information in tables with rows and columns. Each column represents a particular type of information (a field) and each row is one record. Tables are then related to one another by using a common field. A unique field called a key is used to identify each record in a relational database.

You can link to other tables with foreign keys, as long as that table has a primary key.

Advantages of relational multi-table databases

- less redundancy
- Avoidance of inconsistency
- Efficiency
- Data integrity
- Confidentiality

Rules for relational database tables

- Each table has a distinct name
- Each table may contain multiple rows
- Each table has a value to uniquely identify the rows
- Each column in a table has a unique name
- Entries in columns are single values
- Entries in columns are of the same kind (datatype)
- Order of rows and columns is insignificant

Slides 2_2

What is a conceptual model?
Captures the functional and informational needs of a business, is based on current needs but may reflect future needs, addresses the needs of a business (what is conceptually ideal) but does not address its implementation (what is physically possible)

identifies
Important entity (objects that become tables in datavase)
Relationships among entities

Does not specify
Attributes (objects that become columns or fields in database)
Unique identifiers (attribute that becomes primary key in database)

What is a logical model
- Includes all entities and relationships among them
- Is called an entity relationship model (ERM)
- Is illustrated in an ERD (diagram)
- Specifies all attributes and UIDs for each entity.
- Determines attribute optionality
- Determines relationship optionality and cardinality

What is a physical model
- Is an extension to a logical data model
  - Defines table definitions, Data types and precision
  - Identifies views, indeed and other database objects
- Describes how the objects should be implemented in specific database
- Shows all table structures, including columns, primary keys and foreign keys.

Steps to create a physical data model

1. Model entities as tables
2. Model relationships as foreign keys
3. Model attributes as columns
4. Modify the physical data based on physical constraints and requirements,

Conceptual and physical models

- the art of planning developing and communicating produces a desired outcome
- Data modeling is the process of capturing the important concepts and rules that shape a business and depicting them visually in a diagram
- The diagram becomes the blueprint for designing the physical thing
- The clients dream becomes a physical reality

Slides 2_3

Entity
- information that must be tracked
- Name for things that you can list, usually in noun form

Examples: Guest, Room, Reservation, Hotel

Entity Types (Teacher indicated this as extra important)

Name	Description	Example
Prime	Exists independently	CUSTOMER, INSTRUCTOR
Characteristic	Exists because of an other (prime) entity	ORDER, CLASS OFFERING
Intersection	Exists because of two or more entities	ORDER ITEM, CLASS ENROLLMENT

Entities and instances

- Entities contain instances
- An entity instance is a single occurrence of an entity
- Entities represent a set of instances that are of interest to a particular business

Entity	Instance
PERSON	John Smith
PRODUCT	2.5 x 35 copper nail
PRODUCT TYPE	Nail
JOB	Violinist

Attributes
- attributes describe entities and are the specific information that must be known
- It is a single valued property detail of an entity

Example: Age, address, name, phone number

Attribute characteristics
- Attributes are shown within the entity box on the ERD
- Attribute names are singular and mixed case or lowercase.
- In most cases the name of the attribute should not include the entity's name because attributes are qualified with the entity name.
- Attributes are classified as one of the following
  - Mandatory (nulls are not allowed) Indicated by a *
  - Optional (nulls are allowed) Indicated by a o

Volatile and nonvolatile attributes

Volatile attributes are unstable attributes, such as age.
Nonvolatile attributes are stable attributes, such as Birthdate.

Single and composite attributes
- Single or "atomic" attributes are attributes that cannot be divided into subparts
  - Such as: ID
- Composite attributes are attributes that can be divided into smaller subparts that represent basic attributes with independent meaning of their own
  - Such as: First name, middle name, last name

Single-valued and multi-valued attributes
- single valued attributes can have a single value at a particular instance of time
  - example: Student last name
- Multi valued attributes can have more than one value at one time
  - example: address

It's best practice to use single valued/atomic attributes.

Slides 2_4

Unique identifiers

- A unique identifier is an attribute of an entity that meets the following rules
  - It is unique across all instances of the entity
  - It has a non-NULL value for each instance of the entity for the lifetime of the instance
  - It has a value that never changes for the lifetime of the instance
- A UID is a special attribute or group of attributes that uniquely identifies a particular instance of an entity

Example single-attribute ID: ID(PK)
Example multiple-attribute ID: Performance date, seat number
Multi attribute ID's are rarely used.

Simple UID vs Composite UID
- A UID that is a single attribute is a simple UID
- Sometimes a single attribute is not enough to uniquely identify an instance of an entity
- If the UID is a combination of attributes its called a composite UID

Simple: Ticket Number
Composite: Performance Date, Seat Number

Artificial unique identifier

- An artificial UID is made from data that is assigned or generated by the system
- Artificial UIDs do not occur in the natural world but are created for identification purposes in a system

Example: Primary key ID

Candidate unique identifiers

- An entity can have more than one UID
- Candidate UIDs
  - Badge number
  - Payroll number
- Only one of the candidate UIDs can be chosen as the primary UID
- The other candidates are called secondary UIDs

Email is a bad UID, because it can change.

Primary key

- A UID becomes a PK (Primary Key) when the logical model is transformed into a physical database
- A PK is a column or set of columns that uniquely identifies each row in a table
- It cannot contain null values
- A PK is either an existing table column or a column that is specifically generated bu the database according to a defined sequence
- It must contain a unique value for each row of data

Slides 2_5

Relationships
A relationship is bidirectional, significant association between two entities or between an entity and itself

[employees] - - - contain. | Assigned to —< [employee]

- Relationships represent an association between two or more entities
- The relationship line is either solid which means mandatory or dashed, which means optional
- These lines termite in either a single toe, which means one instance, or a crows foot, which means one or more instance.
- Relationships have names that help describe the connection between entities
- In the relationship diagram the name of the relationship, from either perspective, is printed near the starting point of the relationship line.

Examples: Departments CONTAINS employees | Employees ASSIGNED TO departments

Foreign Key
Relationships in a conceptual data model are mapped to foreign keys (FK) in a physical database table

A FK is a column or combination of columns in one table that refers to a PK in the same table or another table

Components of a relationship (teacher indicated this as extra important)

The components of the relationship include the following

- Name: The label that appears close to the entity it is assigned to. Make sure that all relationship names are in lowercase
- Cardinality: The minimum and maximum values in a the relationship
  - One and only one matching records
  - One or more matching records
- Optionality: Whether the relationship must exist
  - Optional (zero matching records)
  - Mandatory (at least one matching record in each entity)

When reading the business rule sentence, use the following working

- optional: use may be or may
- Mandatory: use must be or must
- Line: use one and only one
- Crows feet: use “one or more”

Business rule syntax is as follows

Each entity1 {must be or may be} relationship name {one or more or one and only one} entity2

What is optionality in a relationship?

- Relationships are either mandatory or optional
- Consider the two entities, employee and job
- Based on what you know about instances of the entities, you can determine the optionality by answering two questions
  - Must every employee have a job? In other words, is this a mandatory or optional relationship for an employee?
  - Must every job be done by an employee? In other words, is this a mandatory or optional relationship for a job?

What is cardinality in a relationship?

- Cardinality measures the quantity of something
- In a relationship, it determines the degree too which one entity is related to another by answering “how many”

Relationship types
All relationships represent the information requirements and the rules of the business

- Many to one M:1
- One to many 1:M
- Many to Many M:M - SHOULD NOT BE USED. Make an intersection table for this.
- One to one 1:1

Many to one & One to many relationships
Many to one and one to many relationships have cardinality to one or more in one direction and one and only one in the other direction.

Many to many relationships
Many to many relationships have cardinality of one or more in both directions.

One to one relationships
One to one have cardinality of one and only one in both directions

Recursive relationships
A recursive relationship is a relationship with an entity and itself
Example:each employee may manage one or more employees,.. employee must be managed by one and only one employee.

Slide 2_6

Purpose of conceptual modeling
Should be implementation free (can be used by every type of database model)
Provides a basis for a blueprint.
You understanding the terms helps make a design into a reality
Can be used to further discussion between designers, database administrators and application developers.

It's important because

- describes the exact information needs of the business
- Facilitates discussion
- Prevents mistakes and misunderstandings
- Forms a sound basis for physical database design
- Documents the process (business rules) of the business
- Takes into account regulations and laws governing the industry.

A conceptual model is a formal model which

- describes the things of significance to an organization (entities)
- Identifies the highest level relationships between the different entities but may or may not include the cardinality and nullability
- Does not specify the attributes or the unique identifier for each entity

Logical modeling
The logical model describes the data with as much detail as possible without regard to how it will be implemented

Is usually derived from the *conceptual* model. It includes all entities, attributes, UIDs and relationships as well as optionality and cardinality of these items.

The logical model is illustrated using and ERD.

Steps to create a logical model
1. Identify entities
2. Identify attributes including optionality
3. Identify unique identifiers
4. Determine relationships including optionality and cardinality

Entity relationship diagram
An ERD is a model that identifies the concepts or entities that exist in a system and the relationships between those entities

The database analyst/designer gains a better understanding of the information to be contained in the database through the process of constructing the ERD.

It serves as a documentation tool and is used to communicate the logical structure of the database to users. In particular, it effectively communicates the logic of the database to users.

A graphical representation of entities and their relationships to each other

Goals of ER modeling

- Capture all required information
- Ensure that information appears only once
- Model no information that is derivable from other information that is already modeled
- Locate information in a predictable, logical place

Steps to build an ERD

1. Create entities and attributes
2. Choose unique identifiers
3. Build relationships
4. Identify optionalities and cardinalities
5. Check the model

ERDish
ERDish is the vocabulary used to clearly communicate the business rules that are captured on an ERD.

Components of ERDish

- EACH
- Entity A
- OPTIONALITY (must be/may be)
- RELATIONSHIP (one and only one/one or more)
- Entity B

Slides 3_1

Identifying relationships with multiple entities
An entity can be uniquely identified through multiple relationships

For example the UID for work assignment is employee_ID and Project_Number.
Employee > Work Assignment < Project
This is always a M:M relationship

Use intersection table

M:M Relationships
Attributes describe only entities.
If attributes describe a relationship, the relationship must be resolved

In practice you would make a composite table.

Resolving M:M relationships
Resolving a M:M relationship with a new intersection entity and two 1:m barred relationships.
UID of order item is ORDER number and PRODUCT ID

Intersection entity characteristics

The relationship from the intersection entity are always mandatory.
Intersection entities usually contain consumables like quantity used and dates. They tend to be high volume and volatile entities.
An intersection entity is identified by its two origination relationships (identifying relationships)

Non-transferable relationships
Transferability is the ability of the relationship between two instances of an entity to change over time. A non-transferable relationship cannot be moved between instances of the entity it connects.
A non transferable relationship is represented by a diamond on the relationship
Non-Transferable relationships can *only* be mandatory
example: A membership cannot be moved to another person

Supertype and subtype entities
A supertype has a Parent Child relationship with one or more subtypes.

Subtype is a subgrouping of the entity in an entity type which has attributes that are distinct from those in other subgroupings.

Model the differences between the two subtypes.

When is a relationship applicable: When you can write “Is a” and if the supertype is 100% applicable to the subtype.

100% of all entities in person can be used for a student. Same goes for employees. They all need a goal that’s inherited from the parent.

Everything extra that is not a common given between every child is what belongs to that child. Like for student “Classes taken” and for teachers “classes taught”

Within students you can have more child classes. Like secondary, elementary, higher education, within higher education you can have associates, bachelors and masters. This prevents the ability for an elementary student to be enrolled in a masters program. You can also use it to have a history of events through historical data.

Nobody is the “supertype” but in a subtype. Nobody is a “person” but you are either a student or a teacher. You can’t define all of the edge cases. So you will make an “other” subclass. This is used to dump everything that doesn’t fit within the predefined structure.

Every parents needs at least two children. One of which is “other”

Drawing a subtype
Each subtype is a specialization of a super type and therefore must be enclosed with an entity. The common attributes and relationships for all subtypes must be listed only win the supertype, but they are inherited in every subtype

A subtype can and would generally have attributes or relationships of its own, there can never be just one subtype; another subtype should be created to contain the rest.

Characteristics of a subtype
- Inherits all attributes of a supertype
- Inherits all relationships of the super type
- Usually has its own attributes or relationships
- Is drawn within the super type
- Never exist alone
- May have subtypes of its own
  - Like managers can have a car, but not all employees. So the subtype manager will have its own relationship to “cars”
- Has identical primary keys of the super type and subtype

Example of subtype: ROOM > STANDARD, CLUB, SUITE.

Expiring types
The moment you make a subtype, the parent no longer exists as a type, you give it another child called “other” to catch exceptions.

Generalization and specialization

Generalization is a bottom-up approach where two or more lower level entities are combined to form a higher level entity based on the common features.
VEHICLE as superclass and CAR, TRUCK As a subtype

Specialization is a top down approach where the higher level entity is broken down into low-level entities.
EMPLOYEE as a superclass and current-employee, ex-employee as a subtype

Entity Subtype rules

EXHAUSTIVE

- every instance of a super type is also an instance of one of the subtypes
- OTHER should be included as a subtype to categorize instances the are not defined by one of the existing subtypes

Example: an employee must be full time, part time or other

MUTUALLY EXCLUSIVE
- Every instance of the super type is of one and only one subtype

Example: An employee cannot both be full time and part time

Identifying subtypes correctly
- Is this subtype a kind of super type?
- Have I covered all possible cases? (exhaustive)
- Does each instance fit into one and only one subtype (mutually exclusive)

Nested Subtypes
You can nest subtypes
example: you can nest a super type “staff” in a type “hotel” together with “Non-staff”

Modeling hierarchical data
In a hierarchy you pass down through types.

Recursive relationships
A recursive relationship is always modeled with a loop. Its a relationship where an entity instance is related to another instance in the same entity.

M:M is solved with an intersection table that goes back to itself.

Barker notation
You write a vertical line to indicate that the PK is included inside the table**Arc Relationship**
An art is an exclusive relationship group, which is defined such that only one of the relationship can exist for any instance of an entity.

The art drawn between two relationships connects them and demonstrates mutual exclusivity. The relationship implies an “OR” condtion.

The art indicates that any instance of that entity can have only one valid relationship of the relationship in the arc at any one time.

Characteristics of an arc relationship
The relationships in an arc frequently have the same relationship name.
The relationships in an arc must be either all mandatory or all optional and should have the same cardinality
An entity may have multiple arcs, but a specific relationship can participate only in a single arc.
An arc relationship os represented as the arc shaped line across two or more relationship lines.
Relationships included in the arc have a circle on the relationship arc line.

Slides 3_2

Model data over time: accommodating historical data
- Most businesses need to track *some* historical data to help them find trends and patterns
- Every update of an attribute or transfer of a relationship means loss of information.
  - Not all information is equally useful
- Some systems are required to keep an audit trail of each transaction
- Validate any requirements for storing historical data with the user.
- Storing unnecessary historical data can be costly

You can create additional entities to track the attributes values over time. Like adding an entity to store the status of a contact over time

You can also create additional entities for relationships that may change over tine.
example: rental entity to show the rental relationship over time. Jobs held at a company,

Slides 3_3
Procedural rules are restrictions, such as “A customer who has yet to return a late book can’t loan out a new book” and “an online store might not accept a next day delivery order if the order is received after 3:00PM”

Normalization
The process of organizing the attributes and tables of a relational database to minimize redundancy.

Helps in handling insert, update and delete anomalies, ensuring a better performance of the database.

Why normalize data?
- Reduce redundant data in the design
- Better data integrity
- Design stability
- Eliminate inconsistencies
- Identify missing tables, columns and constraints.

In a nutshell: There is NO duplicate content.

Rule	Description
First Normal Form (1NF)	*All attributes MUST be single-valued.*
Second Normal Form (2NF)	*An attribute must be dependent on its entity's entire UID.*
Third Normal Form (3NF)	*No non-UID attributes can be dependent on another non-UID attribute.*

These are a bit abstract to understand. If you're having trouble with it, in the slides there are practical examples and they make a lot of sense.

First Normal Form (1NF)
1NF Requires that NO multi-valued attributes exist.
You can validate this by checking if each attribute has a single value for each instance of the entity.

If an.attribute is multi-valued you have to. Create an additional entity and relate it to the original entity with a 1:M relationship.

Second Normal Form (2NF)
2NF Requires that any non-UID attribute be dependent on the entire UID. (or be a property/ characteristic of). If the UID is a composite UID each attribute MUST be dependent on ALL parts of the composite UID.

If an attribute is not dependent on the entire UID you must create an additional entity with the partial UID.

Third Normal Form (3NF)
3NF requires that no non-UID attribute can be dependent on another non-UID attribute. It prohibits transitive dependencies

A transitive dependency exists when any attribute in an entity is dependent on any other non-UID attribute in that entity. You must move any non-UID attribute that is dependent on another non-UID attribute into a new entity.

Slides 3_4

Slide 3_4 has a lot of terminology with good examples. Its better if you read that one than read a summary to be honest. I'm not going to bother with mapping or notations.

Terminology mapping

ERD	Physical Design
Entity	Table
Instance	Row
Attribute	Column
Primary UID	Primary Key
Secondary UID	Unique Key
Relationship	Foreign Key

Naming Restrictions

Table and column names
- Must start with a letter
- Can contain up to 30 alphanumeric characters
- Cannot contain spaces or special characters such as "!" But "$", "#" and "_" are permitted

Some words are reserved by oracle DB and SQL. You can't use those either.

Table names MUST be unique within one user account in the oracle database.
Column names must be unique within a table.