Nugget 1: Introduction to Database System

Persistent University

Key Learning Points

1. **Database Design**
2. **Data Dictionary**

Database Management System

- **What is Database?**

  A database is simply an organized collection of related data, typically stored on disk, and accessible by many concurrent users. Databases are managed by a DBMS.

- **What is DBMS?**

  A Database Management System (DBMS) is a set of programs that manages any number of databases.

  A DBMS is responsible for:

  - Accessing data
  - Inserting, updating and deleting data
  - Security
  - Integrity, facilitated by locking, logging, constraints
  - Facilitating backups and recoveries
  - Optimizing performance, etc.

Database Management System

- There are some traditional types of database management systems; which are hierarchical, relational, and network and object relational. Here, we will go in depth of Relational Database Management System.
- Concept of Relational Database:
  - Dr. E. F. Codd proposed the relational model for database system in 1970. It is the basis for Relational Database Management system (RDBMS). The relational model contains the following components
    - Collection of objects or relations
    - Set of operations to act on the relations
    - Data accuracy for integrity and consistency.
- So, Relational Database is basically a collection of Relations or two-dimensional tables.

Persistent

Relational Database Terminology

Concept	Description
Table	A Table is a basic storage structure of an RDBMS, consisting of one or more columns and zero or more rows.
Row	A Row is a combination of column values in a table. It is also referred to as RECORD or TUPLE.
Column	A Column represents one kind of data in a table. It is also referred to as ATTRIBUTE of an entity.
Field	At the intersection of Row and Column, we can find a Field. Field contains the actual data. If there is no data in the field, it is said to contain null value.
Primary Key	A Primary Key is a column or set of columns that uniquely identifies each row in a table. It must contain a value.
Foreign Key	A Foreign Key is a column or set of columns that refers to a primary key in the same or another table. A foreign key value must match the related existing primary key value or else be null.

Relational Database Terminology

- Example:

  Table Name: Customer

customer_id	customer_name	customer_street	customer_city	account_number
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-101
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-201
677-89-9011	Hayes	3 Main St.	Harrison	A-102
182-73-6091	Turner	123 Putnam St.	Stamford	A-305
321-12-3123	Jones	100 Main St.	Harrison	A-217
336-66-9999	Lindsay	175 Park Ave.	Pittsfield	A-222
019-28-3746	Smith	72 North St.	Rye	A-201

- Table: Customer
- Columns: customer_id, customer_name, customer_street, customer_city, account_number
- Primary Key: customer_id

Persistent

Relational Database Properties

- A Relational Database:
  - Can be accessed and modified using Structured Query Language (SQL) statements.
  - Contains a collection of tables with no physical pointers.
  - Uses set of operators

- In a relational DB, user does not need to specify the access route to the table or user does not need to know how the data is arranged physically. User just need to execute structured query language statements, Which is American National Standard Institute (ANSI) standard language for operating relational databases. This language contains large set of operators for combining and partitioning relations. Database can be modified using SQL statements.

Entity Relationship Model

- Models are the cornerstone of any design. Engineers build a model of a car to work out any details before putting into production. In the same way, system engineers develop models to explore ideas and improve the understanding of DB design.
- An Entity Relationship Model is an illustration of various entities in a business and the relationship among them. An ER model is derived from business specifications and built during the analysis phase of SDLC.
- Key components of ER Diagram:

Entity

A thing of significance about which information needs to be known.

For eg: Employee, Department, Supplier, etc.

Attribute

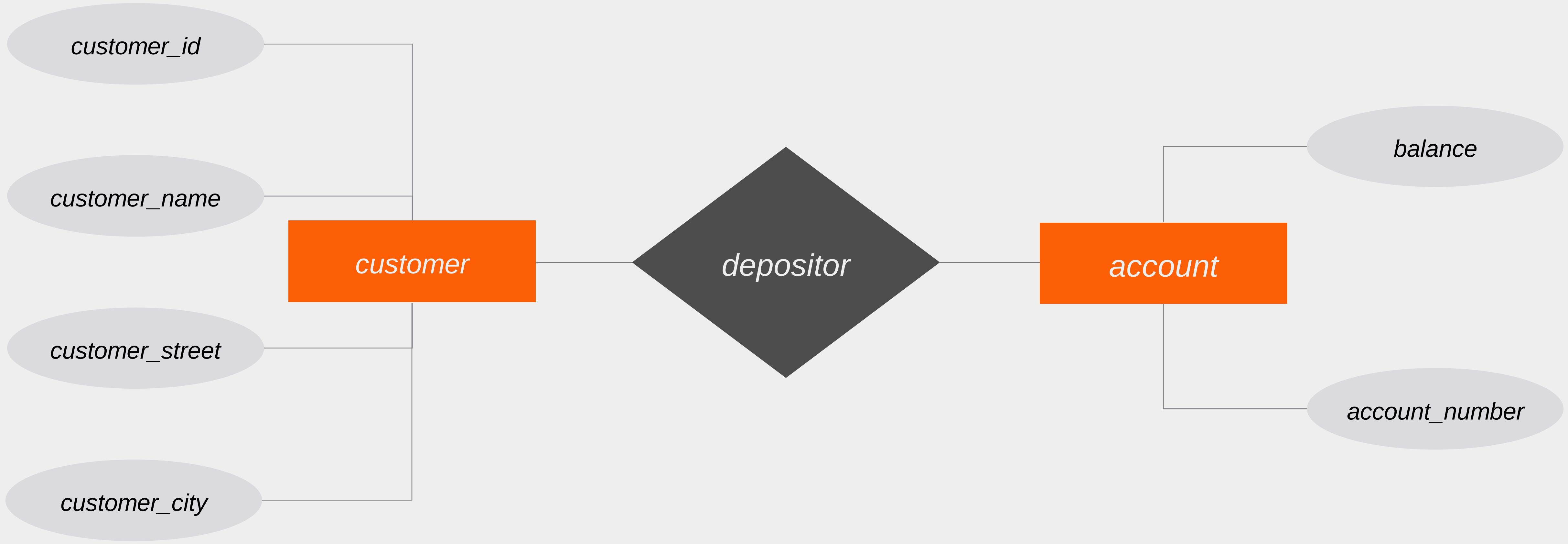
Something that describes or qualifies an entity.

For eg: For entity Employee, attributes could be EmpName, age, gender, title, hire date, etc. Each of the attribute is either required or optional.

Relationship

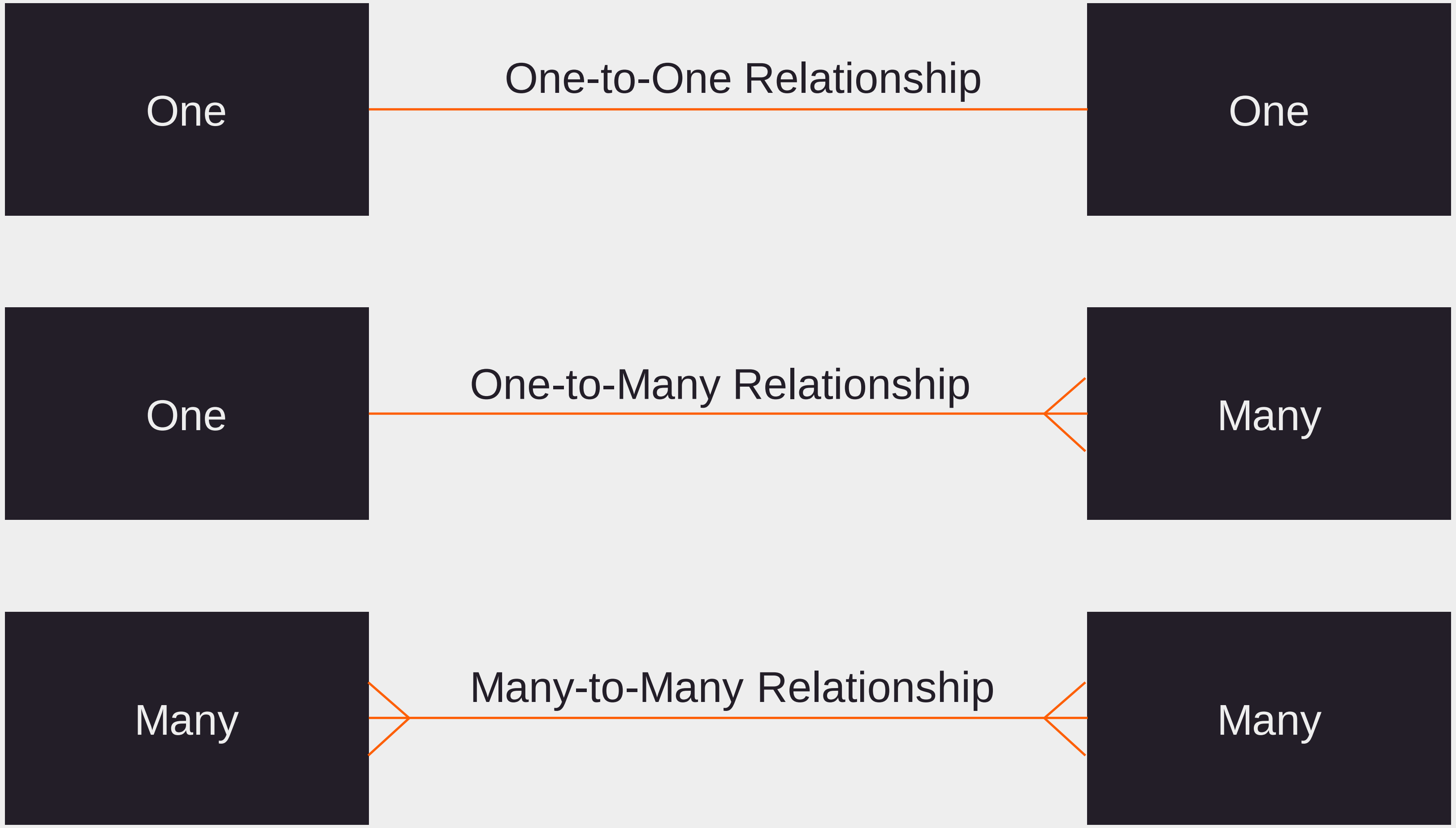
A named association between entities showing optionality and degree/cardinality.

Entity Relationship Model



Types of Relationships

There are different types of relationships between entities.



Types of Relationships

One-to-One Relationship

- Each row in a table is related to a single row in another table.
- For example: Consider these 2 entities:

  address (address_id (PK), address, city, zipcode)

  student (student_id (PK), first_name, last_name, address_id (PK,FK))
- “One student can have one address. One address belongs to one student.”

One-to-Many Relationship

- Each row in a table is related to multiple rows in another table.
- If the primary key in a parent table matches multiple foreign keys in a child table, then the relationship is one-to-many.
- For example: Consider these 2 entities:

  Team (TeamId (PK), TeamName, details)

  Player (PlayerId (PK), PlayerName, country, TeamId (FK))
- “One team contains multiple players. One player belongs to one team only.”

Many-to-Many Relationship

- Each row in each table is related to multiple rows in another table. Each entity may be related to multiple occurrences of the other entity.
- For example: Consider these 2 entities:

  Employee (Emp_id (PK), empName, hire_date)

  Projects (Proj_id (PK), ProjectName, startDate)
- Project contains multiple employees and an employee can belong to one or more projects. In such a case, we can have junction table, which contains the primary keys of both tables.

  employee_proj: (Emp_id (FK), Proj_id (FK))

  Emp_id and Proj_id: Composite PK

Confidential © 2020 Persistent Systems
Persistent

Constraints

Relational databases are composed of relations, managed by relational operations and governed by data integrity constraints.

- Constraints are used to:
  - Enforce rules on the data in a table whenever a row is inserted, updated or deleted. The constraint must be satisfied for the operation to succeed.
  - Prevent the deletion of a table if there are dependencies from other table.
- The following are valid constraint types:

NOT NULL | UNIQUE | PRIMARY KEY | FOREIGN KEY | CHECK | DEFAULT

- Guidelines
  - All constraints are stored in data dictionary. Constraints are easy to reference, if user specifies a meaningful name while creation. If user does not name the constraint oracle server generates a name with SYS_Cn format.
  - Constraints can be created at the time of table creation or after the table has been created.
  - Constraints can be defined at column level or at table level.
  - Constraint details can be viewed using USER_CONSTRAINTS data dictionary table.

Constraints

- NOT NULL Constraint:
  - Not null constraint ensures that column contains no null value. Column without not null constraint can contain NULL values by default.
  - This constraint can be defined only at the column level
  - Column Examples : employee_name, designation, gender

- DEFAULT Constraint
  - The DEFAULT constraint is used to specify default value for a column during an Insert.
  - The default value will be added to all new records, if no other value is specified.
  - Literal values, expressions and SQL functions are allowed default values.
  - Another column name or pseudocolumn can not be used with default.
  - Default data type must match the column data type.

Persistent

Constraints

- UNIQUE Constraint:
  - A unique is an integrity constraint that requires every value in a column or set of columns to be unique – that is no two rows of a table can have duplicate values in a specified column.
  - If the unique constraint comprises more than one column, that set of columns is called as composite unique key. Unique constraints allow input of NULL values unless user also defines NOT NULL constraint. Unique column; without N
  - OT NULL constraint can contain any number of NULLs; as nulls are not considered equal to anything.
  - UNIQUE constraint can be defined at column level or table level.
  - Unique index is automatically created on a column with unique constraint.
  - Column Examples : email_id

Constraints

- PRIMARY KEY Constraint:
  - Primary key constraint creates primary key for a table.
  - It is a column or set of columns that uniquely identifies each row in a table.
  - It is a combination of UNIQUE and NOT NULL constraint i.e. it enforces uniqueness as well as no column that is part of primary key can contain a null value.
  - It can be defined at column level or table level. A composite primary key can be defined by using table level definition.
  - A table can have only one primary key but can have multiple unique constraints.
  - Unique index is automatically created for a primary key column.
  - Column Examples : employeeid, studentid, productid

Constraints

- FOREIGN KEY Constraint:
  - The foreign key constraint or referential integrity constraint, designates a column or set of columns as a foreign key and establishes a relationship with primary key or unique key in the same or a different table.
  - Foreign keys are based on data values and are purely logical; not physical pointers.
  - Foreign key value must match with the existing value in parent table or be NULL.
  - Foreign key constraint can be defined at column level or table level.

Constraints

- CHECK Constraint:
  - It defines a condition that each row must satisfy.
  - The condition can use the same construct as query condition (like condition in WHERE clause)
  - The following expressions are not allowed in the condition
    - References to currval, nextval, ROWNUM, LEVEL pseudocolumns.
    - Calls to USER, SYSDATE, UID and USERENV functions.
    - Queries that refer to other values in other rows.

  - Column Examples :
    - Check gender IN (‘M’, ‘F’, ‘NA’)

      gender data cannot be anything other then M,F or NA
    - Check DOJ <= sysdate

      DOJ cannot be greater than current date / time

Data Dictionary

- A Data dictionary is collection of tables and views that contain the information about the database.
- Data dictionary is created and maintained by Oracle server. All the data dictionary tables are owned by SYS user.
- The base tables are rarely accessed by the user because information in them is not easy to understand. Therefore, users typically access data dictionary views.
- Information stored in data dictionary includes names of the oracle server users, privileges granted to users, db object names, their corresponding details, table constraints etc.

Persistent

Data Dictionary

There are four categories of data dictionary views. Each category has a distinct prefix that reflects its use.

Prefix	Description
USER_	Contains details of objects owned by the User.
ALL_	Contains details of objects owned by the User as well as details of objects to which the user has been granted access rights.
DBA_	These views are restricted views and can be accessed only by people who have been assigned a DBA role.
V$_	Contains information about database server performance, dynamic performance, memory and locking.

Data Dictionary

Some examples of accessing the database information:

- Select table_name from user_tables;
  :- All tables owned by the user
- Select distinct object_type from user_objects;
  :- Distinct object types owned by the user
- Select * from user_constraints;
  :- To view the constraints defined
- Select * from user_views
  :- To see the details of all the views owned by the user
- Select * from user_ind_columns
  :- To view the details of indexed column and corresponding table

Reference Material: Sites

http://www.oracle.com/pls/db102/homepage:

:- Search for required topic in search box on the site.

http://www.dba-oracle.com/concepts/data_dictionary.htm

http://www.smartdraw.com/resources/tutorials/entity-relationship-diagrams/

http://beginner-sql-tutorial.com/sql-integrity-constraints.htm

http://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model

http://www.personal.psu.edu/glh10/ist110/topic/topic07/topic07_05.html

http://docs.oracle.com/javaee/1.2.1/devguide/html/Advanced2.html#11670

Persistent

Session 1: Summary

With this we have come to an end of our first session where we discussed various DBMS concepts.

- At the end of Nugget 1, we see that you are now able to answer following questions:
  - What is DBMS?
  - What is RDBMS?
  - Explain “E-R Diagram and Relationships”
  - What are the types of Constraints?
  - What is Data dictionary and how to use it?

Confidential © 2020 Persistent Systems

Persistent

Thank you!

Persistent University