Lecture Notes – Data Dictionary

Section 1: Lecture Summary

A database contains a **data dictionary** and **data stores**. The **data dictionary** stores table definitions including columns, data types, constraints, relationships like foreign keys, security permissions, indexes, and views. Actual data rows are stored separately in the **data stores**, following the definitions from the data dictionary during insertion and management.

Section 2: Key Concepts and Explanations

Data dictionary holds metadata: table structure (columns, data types, sizes, constraints), inter-table relationships (e.g., foreign keys), user access permissions, indexes, and view definitions as stored SQL. **Data stores** hold the actual data rows (e.g., 5, 5,000, or millions), separate from definitions. The database engine manages these separately, ensuring data insertion adheres to dictionary definitions. Table creation via DDL defines structure stored in the data dictionary; data insertion populates data stores.

Section 3: Example Code and Use Cases

```
-- View Employees table structure (stored in data dictionary)
DESCRIBE Employees;
```

This shows columns like **EmpID int**, **FirstName varchar**, **Salary decimal**, **DeptID int** (foreign key relationship to Departments.DeptID), constraints, and types from companyDB data dictionary.

```
-- Insert data into Employees (stored in data stores, follows dictionary)
INSERT INTO Employees (EmpID, FirstName, LastName, JobTitle, DeptID,
HireDate, Salary)
VALUES (1, 'John', 'Doe', 'Manager', 10, '2023-01-01', 75000.00);
```

Data follows dictionary: **Salary** as **decimal**, **DeptID** validates against Departments.

```
-- Query data stores using dictionary-defined relationships
SELECT e.FirstName, e.Salary, d.DeptName
```

```
FROM Employees e
JOIN Departments d ON e.DeptID = d.DeptID;
```

Section 4: Key Takeaways

Databases separate **data dictionary** (metadata, definitions, relationships, security) from **data stores** (actual rows). DDL like CREATE TABLE populates the data dictionary; data operations use it for validation. The engine handles management complexity.