Tech ABC Corp - HR Database

[Rajat Sharma & 17 Oct 2020]

Business Scenario

Business requirement

Tech ABC Corp saw explosive growth with a sudden appearance onto the gaming scene with their new AI-powered video game console. As a result, they have gone from a small 10 person operation to 200 employees and 5 locations in under a year. HR is having trouble keeping up with the growth, since they are still maintaining employee information in a spreadsheet. While that worked for ten employees, it has becoming increasingly cumbersome to manage as the company expands.

As such, the HR department has tasked you, as the new data architect, to design and build a database capable of managing their employee information.

Dataset

The HR dataset you will be working with is an Excel workbook which consists of 206 records, with eleven columns. The data is in human readable format, and has not been normalized at all. The data lists the names of employees at Tech ABC Corp as well as information such as job title, department, manager's name, hire date, start date, end date, work location, and salary.

IT Department Best Practices

The IT Department has certain Best Practices policies for databases you should follow, as detailed in the Best Practices document.

Step 1

Data Architecture Foundations

Step 1: Data Architecture Foundations

Hi,

Welcome to Tech ABC Corp. We are excited to have some new talent onboard. As you may already know, Tech ABC Corp has recently experienced a lot of growth. Our AI powered video game console WOPR has been hugely successful and as a result, our company has grown from 10 employees to 200 in only 6 months (and we are projecting a 20% growth a year for the next 5 years). We have also grown from our Dallas, Texas office, to 4 other locations nationwide: New York City, NY, San Francisco, CA, Minneapolis, MN, and Nashville, TN.

While this growth is great, it is really starting to put a strain on our record keeping in HR. We currently maintain all employee information on a shared spreadsheet. When HR consisted of only myself, managing everyone on an Excel spreadsheet was simple, but now that it is a shared document I am having serious reservations about data integrity and data security. If the wrong person got their hands on the HR file, they would see the salaries of every employee in the company, all the way up to the president.

After speaking with Jacob Lauber, the manager of IT, he suggested I put in a request to have my HR Excel file converted into a database. He suggested I reach out to you as I am told you have experience in designing and building databases. When you are building this, please keep in mind that I want any employee with a domain login to be have read only access the database. I just don't want them having access to salary information. That needs to be restricted to HR and management level employees only. Management and HR employees should also be the only ones with write access. By our current estimates, 90% of users will be read only.

I also want to make sure you know that am looking to turn my spreadsheet into a live database, one I can input and edit information into. I am not really concerned with reporting capabilities at the moment. Since we are working with employee data we are required by federal regulations to maintain this data for at least 7 years; additionally, since this is considered business critical data, we need to make sure it gets backed up properly.

As a final consideration. We would like to be able to connect with the payroll department's system in the future. They maintain employee attendance and paid time off information. It would be nice if the two systems could interface in the future

I am looking forward to working with you and seeing what kind of database you design for us.

Thanks,
Sarah Collins
Head of HR

Data Architect Business Requirement

- **Purpose of the new database:**

  What is the business partner requesting

  Ans: Business partner experienced a explosive growth with its new AI powered video game console. Due to this, their small team of 10 increased to 200, as well as they started operation in 4 other locations in a year. Now HR is having trouble keeping up with the growth. It has become increasingly cumbersome to manage as the company expands. Therefore business partner is requesting to design and build a database capable of managing their employee information.

- **Describe current data management solution:**

  What is the current method data storage/management
  Ans: Currently all the employee information is maintained on a shared spreadsheet.

- **Describe current data available:**

  What data does the business currently have available
  Ans: The business has an excel workbook consisting of 206 Records, with 11 columns. The data is in denormalized form. The data contains information such as job title, department, Manager's name, hire date, start date, end date, work Location, and salary.

Data Architect Business Requirement

- **Additional data requests:**

  Does the user have future data requests
  Ans: Integration of employee attendance and paid time off information with the database can be done in the future by incorporating the payroll department's system.

- **Who will own/manage data**

  What department will own / manage the data in the database
  Ans: HR department with some management level employees.

- **Who will have access to database**

  List user types that will have access; also list any restrictions to access.
  Ans:

User Types	Restriction
Employees	Read only access to database No access to salary table
HR	No Restriction

Data Architect Business Requirement

- **Estimated size of database**

  List the size of the database in terms of numbers of rows. Business users often understand row or column size instead of GBs or MBs
  Ans:

Number of rows	206
Number of columns	11

- **Estimated annual growth**

  List any expected growth to the data
  Ans: 20% of expected growth each year for next 5 years

- **Is any of the data sensitive/restricted**

  List any data that may be sensitive or restricted from particular users
  Ans: Yes salary data is very much sensitive and it is to be restricted from employees otherwise they would see the salaries of every employee in the company, all the way up to the President.

Data Architect Technical Requirement

- **Justification for the new database**

  Provide at least two justifications for building a database

  1) Expected growth of 20% a year for the next 5 years, therefore, it is a necessity to move the data from spreadsheet to more manageable database.
  2) Issues like data integrity and data security will get resolved.

- **Database objects**

  List the database objects (tables, views, special procedures) that will be created for the database.
  1) Employee Table
  2) Job Table
  3) Department Table
  4) Location Table
  5) Salary Table
  6) EmployeeStatus Table (serving as mapping table)

- **Data ingestion**

  Select a data ingestion method (ETL, Direct feed, API) based on the information provided.

  Ans: Direct Feed

Data Architect Technical Requirement

- **Data governance (Ownership and User access)**

  Ownership: who will own and maintain the data
  Ans: HR

  User Access: who will and will not have access to the data
  Ans:

User Types	Restriction
Employees	Read only access to database No access to salary table
HR	Write access

- **Scalability**

  Should replication or sharding be used to ensure scalability based on user needs
  Ans: No, replication or sharding will incur cost and currently data might only be accessed by hundreds of people at a time. Thus, this condition is not ideal for any form of replication or sharding.

Data Architect Technical Requirement

- **Flexibility**

  Describe measures taken to ensure future data integration if needed
  1) Presence of some similar entities to ensure the fully functional and meaningful integration.

  2) Maintain similar frameworks for the DBMS used in different departments. This will ensure hassle free integration if needed.

- **Storage & retention**

  Storage (disk or in-memory): check IT best practices document

  Retention: how long does the data have to be kept for?
  Ans: As per the federal regulations, it is to be kept for 7 years.

- **Backup**

  IT Best Practices document lists Backup schedule requirements

Step 2

Relational Database Design

Step 2: Relational Database Design

This step is where you will go through the process of designing a new database for Tech ABC Corp's HR department. Using the dataset provided, along with the requirements gathered in step one, you are going to develop a relational database set to the 3NF.

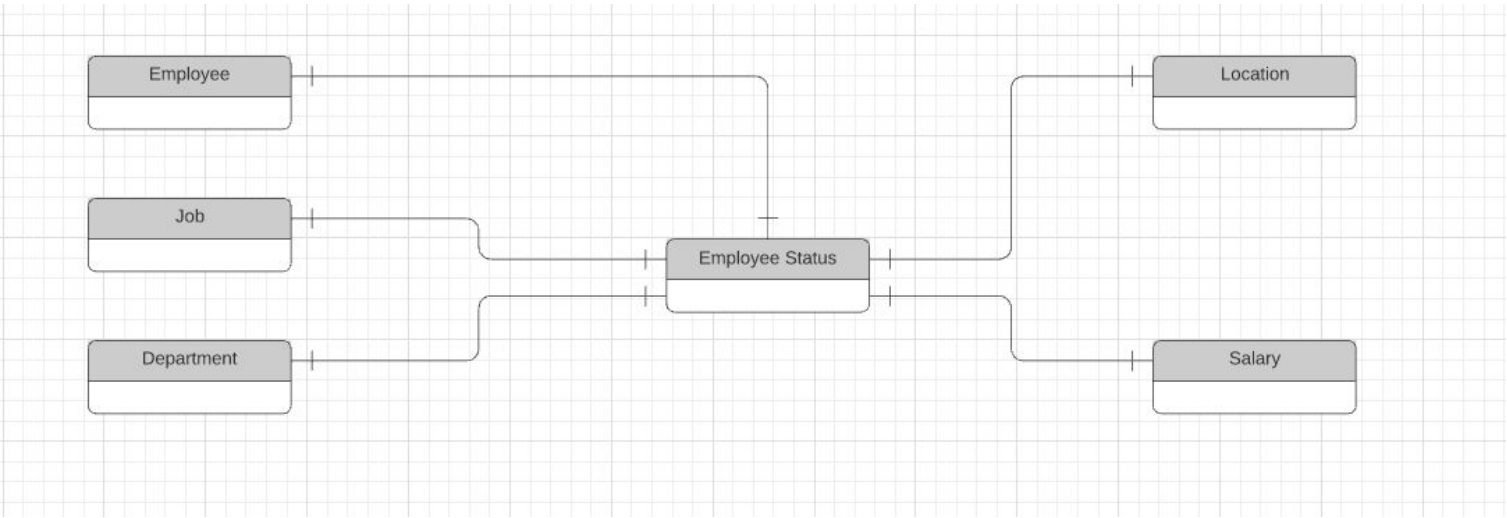
Using Lucidchart, you will create 3 entity relationship diagrams (ERDs) to show how you developed the final design for your data.

You will submit a screenshot for each of the 3 ERDs you create. You will find detailed instructions for developing each of the ERDs over the next several pages.

ERD

- **Conceptual**

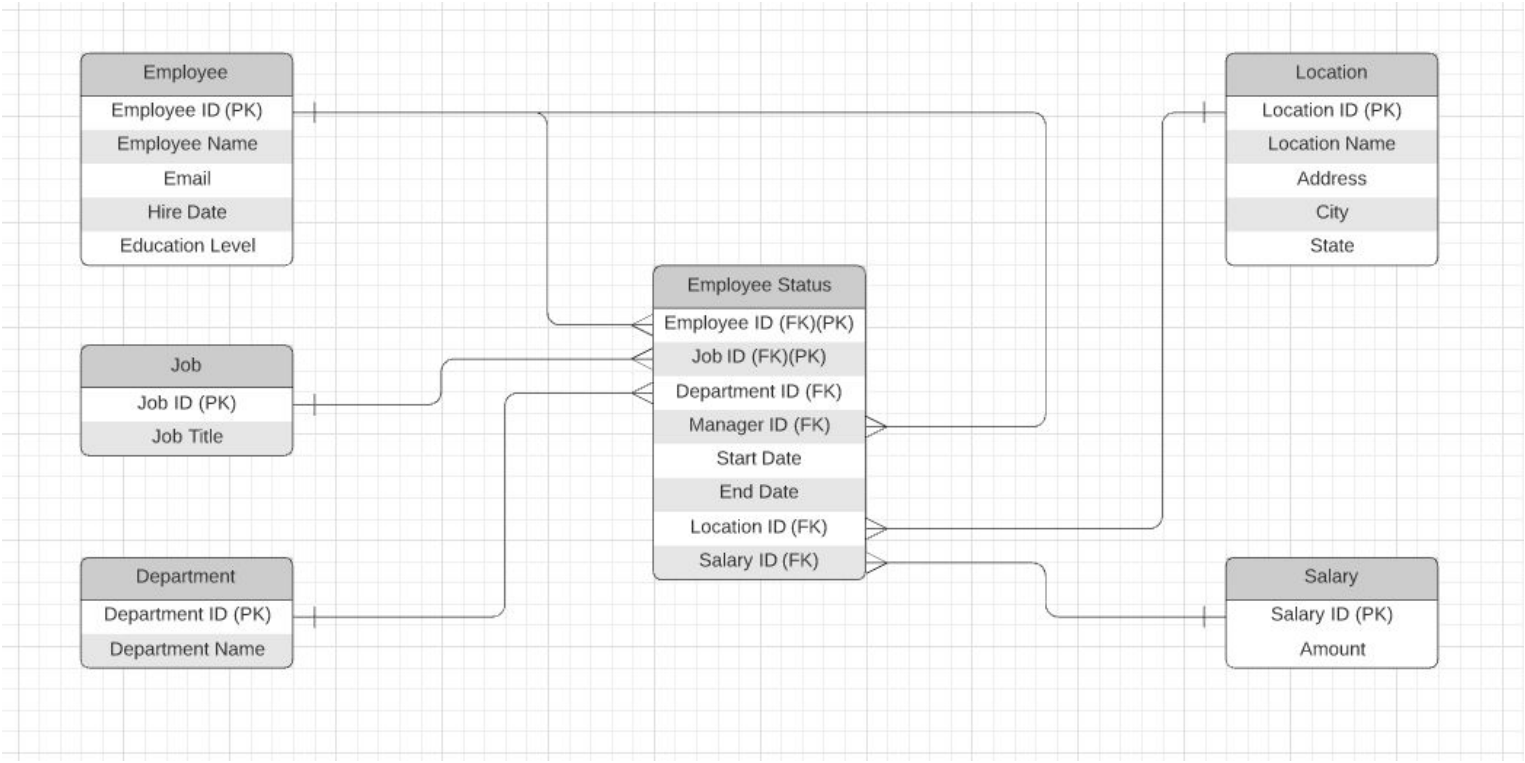
This is the most general level of data modeling. At the conceptual level, you should be thinking about creating entities that represent business objects for the database. Think broadly here. Attributes (or column names) are not required at this point, but relationship lines are required (although Crow's foot notation is not needed at this level). Create at least three entities for this model; thinking about the 3NF will aid you in deciding the type of entities to create.



ERD

- **Logical**

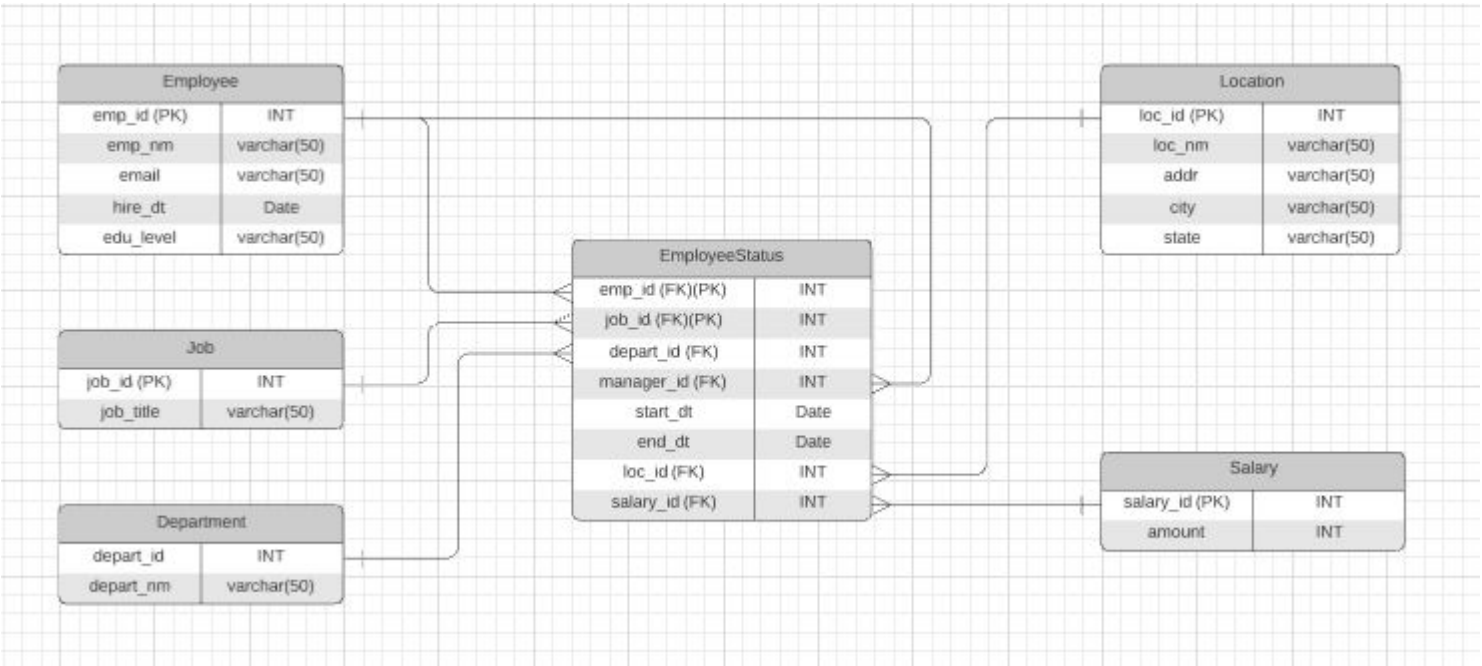
The logical model is the next level of refinement from the conceptual ERD. At this point, you should have normalized the data to the 3NF. Attributes should also be listed now in the ERD. You can still use human-friendly entity and attribute names in the logical model, and while relationship lines are required, Crow's foot notation is still not needed at this point.



ERD

- **Physical**

  The physical model is what will be built in the database. Each entity should represent a database table, complete with column names and data types. Primary keys and foreign keys should also be represented here. Primary keys should be in bold type with the (PK) designation following the field name. Foreign keys should be in normal type face, but have the designation (FK) after the column name. Finally, in the physical model, Crow's foot notation is important.



Step 3

Create A Physical Database

Step 3: Create A Physical Database

In this step, you will be turning your database model into a physical database.

You will:

- Create the database using SQL DDL commands
- Load the data into your database, utilizing flat file ETL
- Answer a series of questions using CRUD SQL commands to demonstrate your database was created and populated correctly

Submission
For this step, you will need to submit SQL files containing all DDL SQL scripts used to create the database.

You will also have to submit screenshots showing CRUD commands, along with results for each of the questions found in the starter template.

DDL

Create a DDL SQL script capable of building the database you designed in Step 2

Hints
The DDL script will be graded by running the code you submit. Please ensure your SQL code runs properly.

Foreign keys cannot be created on tables that do not exist yet, so it may be easier to create all tables in the database, then to go back and run modify statements on the tables to create foreign key constraints.

- Employee Table

```
CREATE table Employee(
    emp_id varchar(10) primary key,
    emp_nm varchar(50),
    email varchar(50),
    hire_dt date,
    edu_level varchar(50)
);
```

- Job Table

```
CREATE table Job(
    job_id serial primary key,
    job_title varchar(50)
);
```

DDL

- Department Table

```
CREATE table Department(
    depart_id serial primary key,
    depart_nm varchar(50)
);
```

- Location Table

```
CREATE table Location(
    loc_id serial primary key,
    loc_nm varchar(50),
    addr varchar(50),
    city varchar(50),
    state varchar(50)
);
```

- Salary Table

```
CREATE table Salary(
    salary_id serial primary key,
    amount int
);
```

DDL

- EmployeeStatus Table

```
CREATE table EmployeeStatus(
    emp_id varchar(10) references Employee(emp_id),
    job_id int references Job(job_id),
    depart_id int references Department(depart_id),
    manager_id varchar(10) references Employee(emp_id),
    start_dt date,
    end_dt date,
    loc_id int references Location(loc_id),
    salary_id int references Salary(salary_id)
);


ALTER table EmployeeStatus
    ADD CONSTRAINT employeestatus_pk
        primary key (emp_id, job_id);
```

CRUD

- **Question 1: Return a list of employees with Job Titles and Department Names**

```
postgres=# SELECT emp.emp_nm, job.job_title, dep.depart_nm
postgres-#     FROM EmployeeStatus as es
postgres-#     JOIN Employee as emp
postgres-#     ON es.emp_id = emp.emp_id
postgres-#     JOIN Job as job
postgres-#     ON es.job_id = job.job_id
postgres-#     JOIN Department as dep
postgres-#     ON es.depart_id = dep.depart_id;
        emp_nm         |        job_title         |      depart_nm
-----------------------+--------------------------+---------------------
 Kumar Durairaj        | Shipping and Receiving   | Distribution
 Kelly Price           | Shipping and Receiving   | Distribution
 Courtney Newman       | Shipping and Receiving   | Distribution
 Prashant Sharma       | Shipping and Receiving   | Distribution
 Jason Wingard         | Administrative Assistant | Distribution
 Michael Sperduti      | Administrative Assistant | Distribution
 Ashley Bergman        | Administrative Assistant | Distribution
 Juan Cosme            | Shipping and Receiving   | Distribution
```

CRUD

- **Question 2: Insert Web Programmer as a new job title**

```
postgres=# INSERT into Job (job_title)
postgres-#     VALUES ('Web Programmer');
INSERT 0 1
postgres=# SELECT * from Job;
 job_id |        job_title
--------+-------------------------
      1 | Manager
      2 | President
      3 | Database Administrator
      4 | Network Engineer
      5 | Shipping and Receiving
      6 | Legal Counsel
      7 | Sales Rep
      8 | Design Engineer
      9 | Administrative Assistant
     10 | Software Engineer
     11 | Web Programmer
(11 rows)
```

CRUD

- **Question 3: Correct the job title from web programmer to web developer**

```
postgres=# UPDATE Job
postgres-#     SET job_title = 'Web Developer'
postgres-#     WHERE job_title = 'Web Programmer';
UPDATE 1
postgres=# SELECT * FROM Job;
 job_id |        job_title
--------+-------------------------
      1 | Manager
      2 | President
      3 | Database Administrator
      4 | Network Engineer
      5 | Shipping and Receiving
      6 | Legal Counsel
      7 | Sales Rep
      8 | Design Engineer
      9 | Administrative Assistant
     10 | Software Engineer
     11 | Web Developer
(11 rows)
```

CRUD

- **Question 4: Delete the job title Web Developer from the database**

```
postgres=# DELETE from Job
postgres-#     WHERE job_title = 'Web Developer';
DELETE 1
postgres=# SELECT * from Job;
 job_id |        job_title
--------+-------------------------
      1 | Manager
      2 | President
      3 | Database Administrator
      4 | Network Engineer
      5 | Shipping and Receiving
      6 | Legal Counsel
      7 | Sales Rep
      8 | Design Engineer
      9 | Administrative Assistant
     10 | Software Engineer
(10 rows)
```

CRUD

- **Question 5: How many employees are in each department?**

```
postgres=# SELECT dep.depart_nm, count(emp.emp_nm)
postgres-#     FROM EmployeeStatus as es
postgres-#     JOIN Employee as emp
postgres-#     ON es.emp_id = emp.emp_id
postgres-#     JOIN Department as dep
postgres-#     ON es.depart_id = dep.depart_id
postgres-#     GROUP BY dep.depart_nm;
      depart_nm      | count
---------------------+-------
 IT                  |    54
 Product Development |    70
 HQ                  |    13
 Distribution        |    27
 Sales               |    41
(5 rows)
```

CRUD

- **Question 6: Write a query that returns current and past jobs (include employee name, job title, department, manager name, start and end date for position) for employee Toni Lembeck.**

```
postgres=# SELECT emp.emp_nm, job.job_title, dep.depart_nm,
postgres-#        (SELECT emp_nm from Employee WHERE emp_id = es.manager_id),
postgres-#        es.start_dt, es.end_dt
postgres-#     FROM EmployeeStatus as es
postgres-#     JOIN Employee as emp
postgres-#     ON es.emp_id = emp.emp_id
postgres-#     JOIN Job as job
postgres-#     ON es.job_id = job.job_id
postgres-#     JOIN Department as dep
postgres-#     ON es.depart_id = dep.depart_id
postgres-#     WHERE emp.emp_nm = 'Toni Lembeck';
    emp_nm    |       job_title        | depart_nm |    emp_nm    |  start_dt  |   end_dt
--------------+------------------------+-----------+--------------+------------+------------
 Toni Lembeck | Database Administrator | IT        | Jacob Lauber | 2001-07-18 | 2100-02-02
 Toni Lembeck | Network Engineer       | IT        | Jacob Lauber | 1995-03-12 | 2001-07-18
(2 rows)
```

CRUD

- **Question 7: Describe how you would apply table security to restrict access to employee salaries using an SQL server.**

Ans:

1) To restrict employees from accessing the employee salaries can be done by **revoking** employees **access** to the Salary table.

2) We can also create different **views** of the EmployeeStatus table so to abstract the critical data elements from displaying on the front when users access database.

Appendix

Additional Info

[1] Link to the Lucidchart:
https://lucid.app/invitations/accept/623456b7-18be-407f-90ca-3fef10fdc5d3

[2] Link to the Github Repo:
https://github.com/rajatsharma369007/designing-hr-db

[3] Udacity Course Material
https://bit.ly/2H8yIqt

[4] Additional Material
https://www.enterprisedb.com/postgres-tutorials/how-implement-column-and-row-level-security-postgresql
https://stackoverflow.com/questions/14884777/how-to-count-rows-on-joined-table