Hi [Manager's Name],

I wanted to clarify why we implemented the **surrogate key approach** in our **Power BI Composite Model with Aggregation Tables**, as I understand there are some questions about this design choice.

### **Why We Chose This Approach**

Our **composite model does not use traditional foreign keys** for dimension relationships. Instead, we adopted a **hash-based surrogate key approach** to optimize performance, reduce model size, and improve scalability. This method ensures that our aggregation tables remain **highly efficient** while keeping the model structure flexible for future changes.

### **Key Benefits of This Approach**

✅ **Data Compression & Performance Gains**

By **grouping records using surrogate keys**, we significantly **reduce the number of rows in the aggregation table**, which minimizes storage and **improves query performance**.

Instead of storing multiple foreign keys in fact tables, a **single surrogate key (hash) efficiently replaces them**, making joins **faster** and reducing redundancy.

✅ **Optimized Aggregation Behavior in Power BI**

Since our aggregation table is in **Import Mode**, Power BI automatically **prioritizes the pre-aggregated data** for most queries, ensuring **fast report performance**.

When users need granular details, Power BI seamlessly **switches to DirectQuery** on the detailed fact table—maintaining **both speed and flexibility**.

✅ **Simplified Power BI Relationships**

Traditional **star schema models require multiple foreign keys**, which increase the complexity of relationships in Power BI.

By using a **single surrogate key**, our model remains **cleaner, easier to maintain, and more efficient for query execution**.

✅ **Scalability & Schema Evolution**

The hash key approach allows us to **easily accommodate schema changes** (e.g., adding new columns like “D”) without affecting existing relationships.

**Backward compatibility** is maintained, ensuring that **old records remain accessible** even when new attributes are introduced.

✅ **Consistent & Reliable Data Processing**

The surrogate key is generated **at the database level**, ensuring **consistent key assignment** across fact and aggregation tables.

This removes the dependency on Power BI to manage relationships, shifting complexity to **Databricks/ETL**, where it is more efficient.

### **Comparison: Surrogate Keys vs. Traditional Foreign Keys**

|  |  |  |
| --- | --- | --- |
| **Approach** | **Advantages** | **Disadvantages** |
| **Surrogate Keys (Current Approach)** | 🔹 Reduces model size 📉 🔹 Faster aggregation queries ⚡ 🔹 Simpler Power BI relationships 🔗 🔹 Handles schema changes flexibly 🔄 | ⚠️ Requires ETL processing for key generation 🛠️ ⚠️ Hash function must prevent collisions 🔑 |
| **Foreign Keys (Traditional Approach)** | 🔹 Intuitive for relational databases 📚 🔹 Easier to model in Power BI 🖥️ | ⚠️ Increased data duplication 📈 ⚠️ More storage required 🛑 ⚠️ Slower query performance 🚶‍♂️ |

Show more

### **Final Thoughts**

Our current implementation is aligned with **best practices for Power BI Composite Models** that use **aggregation tables**. It ensures we maintain a **high-performance, scalable, and flexible data model**, allowing us to handle large datasets efficiently.

Let me know if you'd like a more detailed breakdown or a walkthrough of our implementation.

Best,  
[Your Name]