常州工学院

《分布式数据库开发》大作业 答辩情况记录表

二级学院：计算机信息工程学院　　班　　级：23 软件四

组长姓名：王凌宇乐　　小组成员：张又琰、周煜

课题名称	基于分布式数据库的数据科学教学系统	
课题类型	☐ 实现+验证型	☐ 基于现有系统的验证实验
	☐ 论文阅读与对比报告	☑ 基于分布式数据库的应用开发

答辩情况记录

分片逻辑是什么样的：

在数据架构的底层，我们采取了物理与逻辑相结合的分片策略来应对海量负载。物理层面，系统完全依托于 Firestore 的 Key-Range Sharding 机制，根据文档主键的字典序自动将数据分布到不同的物理 Tablet 上；针对日志风暴等高频写入场景，我们通过 addDoc 生成随机 UUID，有效避免了“尾部热点”现象，确保流量均匀散列。逻辑层面，则通过 ownerId（用户 ID）进行强制隔离，在 projects 集合的所有查询中嵌入用户身份校验。这种设计巧妙地将全局大表切割为数以万计的“私有逻辑分片”，在契合分布式索引特性的同时，实现了天然的租户屏障。

资源池怎么扩展：

系统的资源池扩展遵循计算与存储分离的原则，以实现不同维度的无限伸缩。存储层利用 Firestore 的 Serverless 特性，当数据量或 IOPS 激增时，底层会自动完成分片分裂与资源调配，实现无感扩容。与之对应，计算层（基于 FastAPI 的 Python 后端）则采用完全无状态的水平扩展模式。通过将计算节点部署在 K8s 集群或 Cloud Run 中，我们可以根据请求量实时增加 Pod 数量，前端请求经由负载均衡分发至任意节点，从而确保系统在面对突发流量时依然稳健。

多租户怎么实现

在多租户实现上，我们选择了一套兼顾运维成本与安全性的共享数据库、独立 Schema 变体方案。所有租户的数据统一存放于物理集合中，但通过严格的行级安全隔离机制进行保护。在数据模型层面，每个文档均绑定唯一的 ownerId；在访问控制层面，利用 Firestore Security Rules 进行强制校验，确保 request.auth.uid 必须与文档的 ownerId 完全匹配。这意味着即使在共用物理存储的环境下，不同租户之间也存在一条无法逾越的数据红线，彻底杜绝了越权访问的可能性。

一致性的代价是什么

基于 CAP 定理的考量，为了优先保障系统的高可用性与分区容错性，我们在绝大多数应用场景下选择了最终一致性。这意味着在网络波动或极高并发下，用户提交代码后的状态更新可能会产生毫秒级的收敛延迟，但这换取了系统永不宕机的韧性与极佳的交互流畅度。唯有在涉及“名额抢占”等对数据准确性有绝对要求的关键环节，我们才会通过 Transactions 临时切换至 CP 模式，通过牺牲一定的响应速度来确保数据的绝对强一致。

怎么同步数据

在数据同步机制上，我们摒弃了低效的传统轮询，构建了一套基于推（Push-based）的实时监听体系。该流程以 Firestore onSnapshot 监听器与 WebSocket 长连接为核心：当 Python 计算节点完成任务并更新数据库状态时，Firestore 的 Change Stream 会立即侦测到变更，并将数据增量（Delta）主动推送到所有订阅了

	该查询的活跃客户端。前端 React 组件在接收到快照后自动触发状态更新与 UI 重绘，从而实现了端到端的毫秒级同步，让用户无需刷新即可实时感知系统状态。
答辩成绩	
记录人签名：王凌宇乐 2025 年 1 月 5 日	