Supabaseプラットフォームの戦略的・技術的分析（2025年第3四半期）

第1章 エグゼクティブサマリー

本レポートは、オープンソースのBackend-as-a-Service（BaaS）プラットフォームであるSupabaseについて、その技術的特性、市場での位置づけ、および主要な競合であるGoogle Firebaseとの比較に焦点を当てた包括的な分析を提供するものです。本分析は、技術選定に関わる意思決定者（CTO、アーキテクト、開発リードなど）が、Supabaseを新規プロジェクトの基盤として、あるいは既存のFirebaseプロジェクトからの移行先として評価するための、深く実用的な洞察を提供することを目的としています。

1.1 主要な分析結果

本レポートにおける主要な分析結果は以下の通りです。

1. アーキテクチャの根本的な違い: SupabaseとFirebaseの選択は、単なるツール選定ではなく、長期的なデータアーキテクチャとベンダー依存に関する根本的な戦略決定です。Supabaseはオープンソースの PostgreSQLを基盤とし、リレーショナルデータモデル（SQL）、データ管理の柔軟性、ベンダーロックインからの解放を重視しています [1]。対照的に、FirebaseはGoogle独自のNoSQLデータベース（Firestore）を中核とし、迅速なプロトタイピングとエコシステム内での利便性を最優先しています [3]。
2. データ分析能力: Supabaseには、FirebaseのStream Collections to BigQuery拡張機能のような、単一でシームレスなネイティブデータウェアハウス（DWH）連携機能は存在しません。しかし、その代わりに柔軟で多様なエコシステムを提供します。小規模な分析であればPostgreSQLを直接利用し、本格的なDWH連携にはサードパーティ製のCDC（Change Data Capture）ツールや、Supabaseが提供するForeign Data Wrappers（FDW）を活用できます [4]。このアプローチはFirebaseより設定が複雑ですが、分析スタックの選択肢と管理権限において優れています。
3. デプロイメントの柔軟性: Supabaseは、成熟したSaaSプラットフォームと、AWSなどのIaaS上での堅牢なセルフホスティングオプションの両方を提供し、デプロイメントにおいてFirebaseより優れた柔軟性を持ちます [7]。これにより、迅速な開発から厳格なデータ主権要件まで、幅広いニー

ズに対応可能です。

4. **2025**年時点での市場での地位: 2025年第3四半期現在、Supabaseは開発者コミュニティから絶大な支持と大規模な資金調達を確保し、特にSQLとオープンソースを重視するスタートアップや開発チームにとって、本番環境で利用可能な信頼性の高いプラットフォームとしての地位を確立しています [9]。
5. **Firebase**からの移行: Firebaseからの移行は、単純な「リフト＆シフト」ではなく、大規模な再設計を伴うプロジェクトです。コストの予測可能性やSQLの強力なクエリ能力といったメリットは大きいものの、データモデルの変換、認証システムの移行、クライアントコードの全面的な書き換えなど、多大な開発工数が必要となります [12]。

1.2 最終的な提言

以上の分析に基づき、以下の提言を行います。

- 新規プロジェクトに対して: 複雑なデータリレーションシップを持つアプリケーション、SQLの表現力を必要とするプロジェクト、または将来的なベンダーロックインを回避する戦略を持つチームにとって、SupabaseはFirebaseよりも優れた選択肢です。初期の学習コストはわずかに高いものの、長期的な拡張性と保守性において大きな利点をもたらします。
- 既存の**Firebase**プロジェクトに対して: 既に大規模に稼働しているFirebaseアプリケーションからの移行は、戦術的な切り替えではなく、長期的な戦略的イニシアチブとして検討すべきです。移行の主な動機が、Firestoreのクエリ能力の限界や予測不能なコストにある場合、そのメリットは再設計にかかるコストとリスクを上回る可能性があります。しかし、移行はバックエンドの全面的な再構築に等しいと認識し、十分なリソースと計画をもって臨む必要があります。

結論として、Supabaseは単なる「Firebaseの代替」ではなく、開発者体験とオープンな技術標準を重視した、PostgreSQL中心の次世代開発プラットフォームとして独自の地位を築いています。その採用は、プロジェクトの技術的要件だけでなく、組織の長期的な技術戦略に基づいて慎重に判断されるべきです。

第2章 コアプラットフォーム分析：Supabase vs. Firebase

SupabaseとFirebaseは、どちらもサーバー管理不要のフル機能バックエンドを提供するBaaSプラットフォームですが、その根底にある哲学とアーキテクチャは根本的に異なります。この章では、両プラットフォームの中核をなすデータベース、認証、ストレージの各機能を詳細に比較分析し、その違いがアプリケーション開発に与える影響を明らかにします。

2.1 根源的な相違：PostgreSQL vs. Firestore

両プラットフォームの最も重要な違いは、その中核となるデータベースにあります。この選択は、データモデリング、クエリの柔軟性、スケーラビリティ、そして開発者体験のすべてに影響を及ぼします。

2.1.1 データモデルと設計思想

Supabaseは、世界で最も信頼されているオープンソースのリレーショナルデータベース管理システム（RDBMS）の一つであるPostgreSQLを基盤としています [1]。これにより、構造化データ、ACID準拠のトランザクション、データの整合性が重視されます [2]。開発者はテーブル、カラム、外部キーを用いて厳密なスキーマを定義し、データの関係性をデータベースレベルで保証します。

一方、FirebaseのCloud Firestoreは、Google独自のNoSQLドキュメントデータベースです [1]。コレクションとドキュメント（JSON形式）で構成され、スキーマレスな性質を持ちます。これは、要件が頻繁に変わる初期段階のプロトタイピングや、非構造化データの扱いに柔軟性をもたらしますが、複雑なデータ関係の整合性をアプリケーション側で担保する必要があります [18]。

この違いは、開発プロセスにおけるトレードオフを生み出します。Firebaseは初期開発のスピードを加速させる「魔法」のような体験を提供しますが、アプリケーションが成長し、データ構造が複雑化するにつれて、その抽象化が逆に足かせとなることがあります [13]。Supabaseは初期段階でスキーマ設計という規律を要求しますが、その規律が長期的なスケーラビリティと保守性の基盤となります [19]。

2.1.2 クエリ能力とパフォーマンス

SupabaseはPostgreSQLの全機能を継承しており、開発者は複雑なJOIN、サブクエリ、集計関数、全文検索、地理空間クエリなどを単一のSQL文で実行できます [1]。これにより、サーバーサイドのロジックをデータベース内に集約でき、アプリケーションコードを簡潔に保つことが可能です。

対照的に、Firestoreのクエリは「シャロー（浅い）」であり、異なるコレクション間の関係を横断するクエリはネイティブにサポートされていません [1]。関連データを取得するには、クライアント側で複数回のクエリを発行して手動で結合するか、あるいはデータを非正規化（重複して保持）する必要があります。この制約は、アプリケーションがスケールするにつれて、パフォーマンスのボトルネックやデータ管理の複雑化を招く主要な要因となります [12]。

2.1.3 リアルタイム機能とオフラインサポート

リアルタイムデータ同期は両プラットフォームの主要機能ですが、その実現方法は異なります。Firebaseのリアルタイム機能は長年の実績があり、非常に堅牢で、優れたオフラインサポートをネイティブで提供します [1]。クライアントSDKは、オフライン時でもデータの読み書きをキャッシュし、接続が回復した際に自動的に同期します。

Supabaseのリアルタイム機能は、PostgreSQLの論理レプリケーション機能（WAL: Write-Ahead Log）を利用してデータベースの変更を検知し、WebSocketを通じてクライアントにブロードキャストします [1]。これは強力で透明性の高い仕組みですが、Firebaseのようなネイティブのオフラインファースト機能は提供していません。堅牢なオフライン体験を実現するには、PowerSyncやElectricSQLといったサードパーティ製の同期レイヤーを別途導入する必要があり、これによりアーキテクチャの複雑性とコストが増加する可能性があります [1]。

2.2 認証サービス：Supabase Auth vs. Firebase Authentication

認証機能はBaaSプラットフォームの要であり、両者は豊富な機能を提供しつつも、セキュリティモデルにおいて重要な違いがあります。

2.2.1 機能セット

Supabase AuthとFirebase Authenticationは、Eメール/パスワード認証、ソーシャルログイン（OAuth）、マジックリンクといった標準的な認証方法を網羅しています [12]。Firebase Authは、特にモバイルアプリケーションにおいて、GoogleやAppleサインインのネイティブUIフローが洗練されており、より polished な体験を提供すると評価されています [22]。Supabase Authは、オープンソースのGoTrueサーバーを基盤としており、透明性とカスタマイズ性に優れています [19]。

2.2.2 セキュリティモデルと認可

両者の決定的な違いは、認可（Authorization）の仕組みにあります。Supabaseの最大の特徴は、

認証システムとPostgreSQLの行単位セキュリティ（Row-Level Security, RLS）が緊密に統合されている点です [1]。RLSを利用することで、「ユーザーは自身が作成したデータにのみアクセスできる」といった複雑なアクセスポリシーを、

auth.uid()のような認証情報を用いてSQLで直接データベースに記述できます。

このアプローチは、認可ロジックをアプリケーション層からデータ層に移行させるパラダイムシフトを意味します。データベース自体がセキュリティポリシーの単一の真実の源泉（Single Source of Truth）となり、クライアントはより汎用的なクエリを発行するだけで、データベースが自動的にアクセス制御を適用します。これにより、アプリケーションコードが簡素化され、セキュリティ体制の監査性と堅牢性が向上します。

一方、Firebaseは、FirestoreとStorageに対してそれぞれ独立したJSONベースのセキュリティルール言語を使用します [25]。これは柔軟ですが、サービス間でルールの一貫性を保つことが複雑になりがちで、認可ロジックが分散する傾向にあります。RLSは非常に強力ですが、設定を誤ると重大なセキュリティインシデントにつながるため、開発者にはSQLとRLSへの深い理解が求められます [26]。

2.3 オブジェクトストレージ：Supabase Storage vs. Firebase Cloud Storage

ファイルやメディアの管理においても、両者はアーキテクチャとアクセスコントロールの点で異なります。

2.3.1 アーキテクチャ

Firebase Storageは、実績のあるGoogle Cloud Storageを基盤としており、高い信頼性とスケーラビリティを提供します [1]。Supabase Storageは、S3互換のオブジェクトストレージであり、その最大の特徴はファイルメタデータ（ファイル名、所有者、MIMEタイプなど）をPostgreSQLの専用テーブルに直接格納する点です [1]。

2.3.2 アクセスコントロール

このメタデータのデータベース格納というアーキテクチャ上の選択が、Supabase Storageに大きな利点をもたらします。ファイルのアクセス権限ポリシーを、データベースのデータと同じ強力なRLSエンジンを用いてSQLで記述できるのです [1]。これにより、「有料プランのユーザーのみが特定のフォルダ

内の動画をストリーミングできる」といった、データベース内のユーザー情報と連携した動的で詳細なアクセスコントロールが容易に実現できます。Firebaseでは、ストレージ用の独立したセキュリティルールでこれを実現する必要があり、データベースとの連携はより間接的になります。

2.4 比較表と総括

以下の表は、SupabaseとFirebaseのコア機能における主要な違いをまとめたものです。

機能	Firebase	Supabase	主要な差別化要因と洞察
データベース	NoSQL (Cloud Firestore)	RDBMS (PostgreSQL)	アーキテクチャの根幹。NoSQLの柔軟性か、SQLの構造と整合性かの選択。
データモデル	スキーマレスなドキュメント	構造化されたテーブル	Firebaseは初期開発が速いが、スケール時に複雑化。Supabaseは長期的な保守性に優れる。
クエリ言語	独自のクライアントSDK	標準SQL	Supabaseは複雑なクエリやデータ分析に圧倒的に強い。Firebaseはクライアント側での処理が多い。
トランザクション	限定的なサポート	完全なACID準拠	Supabaseはデータの整合性が求められる金融系などのアプリケーションに適している。
リアルタイム	成熟したネイティブ機能	PostgreSQLの論理レプリケーション	Firebaseはよりシームレス。Supabaseは強力だが、設定や

			サードパーティツールが必要な場合がある。
オフラインサポート	強力なネイティブ機能	サードパーティ製ツールが必要	モバイルでのオフラインファースト要件が厳しい場合、Firebaseに利点がある。
認証モデル	Firebase Authentication	Supabase Auth (GoTrue)	機能セットは類似しているが、Supabaseはオープンソースである点が特徴。
セキュリティ	独立したセキュリティルール	行単位セキュリティ **(RLS)**	パラダイムシフト。Supabaseは認可ロジックをDBに集約し、より堅牢なセキュリティを実現。
ストレージ	Google Cloud Storage	S3互換 + PostgreSQLメタデータ	SupabaseはRLSによる動的なアクセスコントロールで優位性を持つ。
サーバーレス	Cloud Functions (多言語)	Edge Functions (Deno/TypeScript)	FirebaseはGoogle Cloudとの連携が深い。Supabaseはエッジでの低遅延実行に特化。
オープンソース	No	**Yes**	ベンダーロックインの回避。Supabaseは透明性とデータポータビリティを提供する。
セルフホスト	No	**Yes**	データ主権や特殊なインフラ要件がある

			場合にSupabaseは唯一の選択肢となる。
価格モデル	従量課金制 (読み書き/操作)	リソースベース (DBサイズ/CPU)	Supabaseはコストが予測しやすく、読み取りヘビーなアプリで有利。Firebaseは急激なコスト増のリスクがある。

総括すると、Firebaseは迅速なプロトタイピングとGoogleエコシステムとの緊密な連携を求めるプロジェクト、特にモバイルファーストのリアルタイムアプリケーションに適しています。しかし、その利便性は「複雑さの上限」という代償を伴います。一方、SupabaseはSQLのパワー、オープンソースであることの自由度、そしてデータに対する完全なコントロールを求める開発者にとって、より戦略的で長期的な選択肢となります。初期のハードルはわずかに高いものの、そのアーキテクチャはアプリケーションの成長と共にスケールし、複雑な要件にも対応できる高い「上限」を提供します。

第3章 データ分析とデータウェアハウジング能力

アプリケーションの成長に伴い、蓄積されたデータを分析し、ビジネスインテリジェンス（BI）や機械学習（ML）に活用することは不可欠です。本章では、FirebaseとSupabaseが提供するデータ分析およびデータウェアハウス（DWH）連携の能力を比較し、それぞれのアーキテクチャ上のアプローチとトレードオフを評価します。

3.1 Firebaseの標準：ネイティブBigQuery連携

Firebaseは、Google Cloudエコシステムの一部として、データ分析のための非常に強力かつシンプルなソリューションを提供しています。その中核となるのが、**Stream Collections to BigQuery** というFirebase拡張機能です。

この拡張機能は、Firestoreコレクションへのデータの追加、更新、削除といった変更をリアルタイムで検知し、その変更データをGoogleのサーバーレスDWHであるBigQueryに自動的にストリーミングします。これは実質的に、設定不要のフルマネージドなChange Data Capture（CDC）パイプラインとして機能します。開発者は数クリックでこの拡張機能を有効にするだけで、複雑なETL（Extract,

Transform, Load）パイプラインを構築・管理することなく、アプリケーションデータをほぼリアルタイムで分析できるようになります。このシームレスな統合は、Firebaseの大きな強みの一つです。

3.2 Supabaseの多角的な分析戦略

Supabaseには、FirebaseのBigQuery拡張機能に直接相当する単一のネイティブ機能は存在しません。しかし、その代わりに、オープンなアーキテクチャを活かした複数の選択肢を提供しており、開発者はプロジェクトの要件や既存の技術スタックに応じて最適な分析アーキテクチャを構築できます。

3.2.1 オプション1：PostgreSQLによる直接分析

小規模から中規模のデータセット（目安として最大50万行程度）であれば、SupabaseのPostgreSQLデータベースを直接分析クエリの対象とすることが可能です [27]。このアプローチでは、以下のような標準的なPostgreSQLの最適化技術を活用します。

- インデックス作成: WHERE句で頻繁に使用されるカラムにインデックスを作成し、クエリのパフォーマンスを向上させる [28]。
- マテリアライズドビュー: 事前に集計結果を計算して物理的に保存しておくことで、複雑な集計クエリを高速化する [27]。
- テーブルパーティショニング: 大規模なテーブルを日付や地域などのキーで小さなパーティションに分割し、クエリがスキャンするデータ量を削減する [27]。

しかし、この方法には限界があります。PostgreSQLは本来、トランザクション処理（OLTP）に最適化された行指向データベースであり、大規模データに対する複雑な分析処理（OLAP）には最適化されていません [27]。本番環境のデータベースに対して重い分析クエリを実行すると、アプリケーション本体のパフォーマンスを低下させるリスクがあります [30]。

3.2.2 オプション2：サードパーティツールによる外部DWH連携

より本格的な分析を行うための推奨アプローチは、データをBigQuery、Snowflake、Redshiftといった専用のDWHに複製することです。Supabaseは標準的なPostgreSQLであるため、データエンジニアリングのエコシステムに存在する多くのCDCおよびETL/Reverse ETLツールと互換性があります。

- **CDC**プラットフォーム: Artie [5] や Estuary [31] のようなサービスは、PostgreSQLの論理レプリケーション機能に接続し、Supabaseデータベースへの変更をリアルタイムでDWHにストリーミングし

ます。これは、FirebaseのBigQuery拡張機能に最も近い、技術的に優れた代替手段です。

- ワークフロー自動化ツール: n8n [33]、Zapier [4]、viaSocket [34] といったプラットフォームを利用して、特定のイベント（例：新しい行が挿入された時）をトリガーにデータをDWHに送信するパイプラインを構築できます。これは設定が容易ですが、真のCDCに比べてパフォーマンスやリアルタイム性で劣る場合があります。

3.2.3 オプション3：ネイティブForeign Data Wrappers (FDW)

Supabaseは、PostgreSQLの強力な機能であるForeign Data Wrappers（FDW）の活用に力を入れています。FDWは、外部のデータソースをPostgreSQL内からあたかもローカルテーブルであるかのようにクエリできるようにする拡張機能です。Supabaseは公式に**bigquery_wrapper**を提供しており [4]、これによりSupabaseのライブデータとBigQueryの分析データをSQLの

JOINで結合するといった、データフェデレーション（データ連合）が可能になります。これは非常に強力ですが、パフォーマンスは外部システムに依存するため、完全なETLパイプラインの代替とはなりません。

3.2.4 オプション4：登場しつつあるネイティブソリューション - Analytics Buckets

Supabaseは、分析ワークロードとトランザクションワークロードを分離するためのネイティブソリューションとして、**Analytics Buckets**の開発を進めています [30]。これは、オープンなテーブルフォーマットであるApache Icebergを基盤としており、オブジェクトストレージ上で大規模なデータセットに対する分析を行うことを目的としています [35]。2025年第3四半期現在、この機能はまだアルファ版ですが、将来的により統合された高性能な分析ソリューションを提供するというSupabaseの戦略的な方向性を示しています。

3.3 分析アーキテクチャの選択に関する提言

FirebaseとSupabaseのデータ分析能力を比較すると、明確なトレードオフが見えてきます。Firebaseは、Google Cloudエコシステム内で完結する、**統合型（Integrated）**ではあるもののプロプライエタリな分析パスを提供します。これは非常にシンプルで強力ですが、ユーザーをGoogleの分析スタックにロックインします。

一方、Supabaseは、特定のパスを強制するのではなく、**エコシステム型（Ecosystem-driven）**の

アプローチを取ります。標準的なPostgreSQLの機能を外部に公開することで、開発者は既存のデータエンジニアリングツール群を自由に活用できます。これにより、DWH（Snowflake, Redshift, BigQueryなど）やCDCツールを自由に選択し、自社のニーズや既存インフラに最適なベストオブブリードの分析スタックを構築する柔軟性が得られます。

この柔軟性の代償として、Supabaseで堅牢な分析パイプラインを構築するには、Firebaseを利用する場合よりも高度なデータエンジニアリングの専門知識が必要となります。しかし、それは同時に、ベンダーロックインを回避し、将来にわたって技術選択の自由を確保するための戦略的な投資と捉えることができます。SupabaseがFDWやAnalytics Bucketsのようなネイティブ機能を開発していることは、この複雑さを認識し、将来的により統合された選択肢を提供しようとしている証左と言えるでしょう。

第4章 アーキテクチャの深掘り：Supabase vs. TiDB

Supabaseが提供するPostgreSQLベースのアーキテクチャは、多くのアプリケーションにとって十分なスケーラビリティを持ちますが、インターネットスケールの超大規模トランザクションを扱うシステムでは、異なるアーキテクチャが求められることがあります。本章では、Supabaseのアーキテクチャを、分散SQLデータベースの代表格であるTiDBと比較し、それぞれの設計思想、スケーラビリティ、そして最適なユースケースの違いを明確にします。この比較は、SupabaseとTiDBが直接的な競合製品ではないことを示し、それぞれが解決しようとしている問題領域の違いを浮き彫りにします。

4.1 アーキテクチャの原則

4.1.1 Supabase (PostgreSQLベース)

Supabaseのデータベース層は、伝統的なモノリシック（単一ノード）アーキテクチャのPostgreSQLに基づいています [36]。Supabaseプラットフォーム全体は、認証用のGoTrue、API生成用のPostgRESTなど、複数のマイクロサービスで構成されていますが、データ永続化の中核は単一のPostgreSQLインスタンスが担います [7]。このアーキテクチャのスケーリングは、主に

垂直スケーリング（サーバーのCPUやメモリを増強する）と、読み取り負荷を分散するためのリードレプリカの追加によって行われます [1]。これは数十年にわたって実績のある、堅牢で理解しやすいモデ

ルです。

4.1.2 TiDB (分散NewSQL)

TiDBは、Google Spannerに触発された分散**NewSQL**データベースであり、ゼロから水平スケーラビリティを実現するために設計されています [36]。その最大の特徴は、

コンピュート（計算）層とストレージ層の分離です [37]。

- コンピュート層 **(TiDB Server)**: ステートレスなSQL処理レイヤーであり、クライアントからのクエリを受け付けて解析・実行します。MySQLプロトコルと互換性があります [37]。
- ストレージ層 **(TiKV)**: 分散型のKey-Valueストアであり、データを複数の物理ノードに自動的にシャーディング（分割）して格納します。
- 管理層 **(Placement Driver, PD)**: クラスタ全体のメタデータを管理し、トランザクションのタイムスタンプを割り当てることで、分散環境下での厳密なACIDトランザクションを保証します [37]。

このアーキテクチャにより、TiDBは読み書きの処理能力とデータストレージ容量を、それぞれ独立してスケールさせることが可能です。

4.2 スケーラビリティとパフォーマンス

スケーリングモデルの違いは、パフォーマンス特性に直接影響します。

- **Supabase (垂直スケーリング)**: 負荷が増加した場合、より高性能なコンピュートインスタンスにアップグレードすることで対応します [39]。この方法はシンプルですが、単一サーバーの物理的な性能限界という上限が存在します。書き込み処理はプライマリノードに集中するため、書き込みヘビーなワークロードのスケーラビリティには限界があります。
- **TiDB (水平スケーラビリティ)**: 負荷が増加した場合、TiDBサーバー（コンピュート）やTiKVサーバー（ストレージ）のノードをクラスタに追加するだけで、リニアに性能を向上させることができます [36]。これにより、理論上は無限に近いスケーラビリティを実現できますが、クラスタの運用管理はモノリシックなデータベースよりも複雑で、より多くのリソースを消費します [38]。

両データベースともACID準拠のトランザクションを保証しますが、TiDBは分散トランザクションプロトコルを用いて、地理的に分散したノード間でもデータの一貫性を維持します。これは技術的に非常に高度な達成です [36]。

4.3 ユースケースの明確な違い

以上のアーキテクチャ分析から、SupabaseとTiDBは異なる問題領域を解決するためのツールであり、直接競合するものではないことが明らかになります。

- **Supabase**は**BaaS**プラットフォーム: Supabaseの主な価値は、開発者体験を最大化することにあります。PostgreSQLデータベースに加えて、認証、ストレージ、サーバーレス関数といったバックエンド機能をオールインワンで提供し、開発者が迅速にアプリケーションを構築・スケールさせることを支援します [15]。その対象ユーザーは、MVPから数百万ユーザー規模のサービスまでを目指す
  アプリケーション開発者です [40]。
- **TiDB**は特化型データベースシステム: TiDBの主な価値は、伝統的なRDBMS（MySQLやAuroraなど）のスケーラビリティの限界を超えることにあります。書き込み負荷が極めて高い、ミッションクリティカルな大規模OLTPシステムのために設計されています [41]。その対象ユーザーは、超大規模なEコマース、金融サービス、オンラインゲームなどのバックエンドで、データベースの性能限界に直面している
  データベース管理者（**DBA**）やインフラエンジニアです。

結論として、この比較は「どちらが優れているか」を問うものではありません。Supabaseは「開発者がいかに生産的にバックエンドを構築できるか」という課題を解決し、TiDBは「データベースが物理的な限界を超えていかにスケールできるか」という課題を解決します。Supabaseは、伝統的なRDBMSであるPostgreSQLの力を現代的な開発プラットフォームを通じて最大限に引き出す「開発者体験の未来」を代表しています。一方、TiDBは、RDBMSのコアアーキテクチャ自体をクラウドネイティブな分散環境のために再発明する「インフラストラクチャの未来」を代表していると言えるでしょう。技術選定の際には、自社のプロジェクトが直面している課題がどちらの領域に属するのかを正確に見極めることが重要です。

第5章 デプロイメントモデルと運用の考慮事項

Supabaseは、そのデプロイメントモデルにおいて高い柔軟性を提供しており、これはプロプライエタリなSaaSであるFirebaseに対する大きな差別化要因です。開発者は、フルマネージドのクラウドサービスを利用する手軽さと、自社のインフラ上で運用する完全なコントロール権を両天秤にかけることができます。本章では、Supabase Cloud（SaaS）とAWS上でのセルフホスティングという2つの主要なデプロイメントモデルを分析し、それぞれの利点、欠点、および運用上の考慮事項を詳述します。

5.1 Supabase Cloud：マネージドSaaSオファリング

Supabase Cloudは、公式にサポートされている最も一般的な利用形態です [8]。数分でプロジェクトを立ち上げることができ、インフラ管理の複雑さから開発者を解放します。

- 特徴と利点:
  - 迅速なセットアップ: ウェブダッシュボードから数クリックで、PostgreSQLデータベース、API、認証、ストレージを含む完全なバックエンド環境が利用可能になります。
  - フルマネージド: データベースのバックアップ（有料プランでは日次自動バックアップ）、パッチ適用、スケーリングといった運用タスクはSupabaseチームによって管理されます [28]。
  - 統合された体験: データベース管理UIである「Studio」、ログ監視、請求管理などがすべて単一のダッシュボードに統合されています [43]。
- 価格体系:
  - 価格設定は、Firebaseの操作ごとの従量課金制とは異なり、リソースベースの階層型モデル（Free, Pro, Team, Enterprise）を採用しています [39]。これはデータベースサイズ、コンピュートインスタンスのスペック、ストレージ容量、データ転送量などに基づいており、コストが予測しやすいと評価されています [18]。
- 制約と考慮事項:
  - リージョン制限: プロジェクトは特定のAWSリージョンにホストされるため、ユーザーが地理的に離れている場合、データアクセスに遅延が生じる可能性があります [46]。
  - フリープランの制約: 無料プランのプロジェクトは、1週間の非アクティブ期間後に自動的に一時停止（pause）されます。これを解除し忘れると、プロジェクトが完全に削除されるリスクがあり、注意が必要です [22]。
  - 設定の柔軟性: セルフホスティングに比べ、基盤となるPostgreSQLの詳細な設定やハードウェア構成に対するコントロールは制限されます [46]。

5.2 AWSインフラ上でのセルフホスティング

Supabaseは完全にオープンソースであり、自己のインフラストラクチャ上でホスティングすることが可能です [2]。公式にはDocker Composeを使用したデプロイが推奨されており、これにより必要なすべてのマイクロサービスをまとめて起動できます [7]。

- **AWS**上でのデプロイパターン:
  - **EC2**インスタンスでの直接実行: 最もシンプルな方法は、Amazon EC2インスタンス（例：Ubuntu）を起動し、その上で直接Docker Composeを実行することです [47]。これによりインフラを完全にコントロールできますが、インスタンスの管理、セキュリティ設定、スケーリングなどをすべて手動で行う必要があります。
  - マネージドサービスの活用 **(ECS/Aurora)**: より本番環境に適した構成として、コミュニティ主導でCloudFormationやCDKのテンプレートが開発されています [50]。これらは、Supabaseのステートレスなコンポーネント（APIゲートウェイ、認証サーバーなど）をAWS

Fargate上のAmazon ECSで実行し、データベース層にはマネージドサービスである Amazon Aurora Serverlessを利用します。これにより、スケーラビリティと信頼性が向上しますが、セットアップの複雑性も増大します。

    - **AWS Marketplace**の利用: 2024年4月以降、SupabaseはAWS Marketplaceで提供されています [43]。注意すべきは、これがSupabaseの **SaaS**プロダクトを購入するための調達チャネルであり、ユーザー自身のAWSアカウント内にワンクリックでセルフホスト環境を構築するものではないという点です [43]。ただし、Meetrix.ioなどのサードパーティベンダーが、セルフホスティング用の設定済みAMI（Amazon Machine Image）をMarketplaceで提供している例もあります [49]。
- トレードオフの分析:
    - 利点: セルフホスティングは、究極のコントロールとデータ主権を提供します。SaaSでは対応できない特定のセキュリティコンプライアンス要件（HIPAA, GDPRなど）を満たしたり、アプリケーションサーバーと同じVPC内にデータベースを配置してネットワーク遅延を最小化したりすることが可能です [46]。また、大規模な運用においては、SaaSよりもコストを削減できる可能性があります。
    - 欠点: 最大の欠点は、運用オーバーヘッドの増大です。アップデートの適用、セキュリティパッチの管理、監視、バックアップ戦略の策定と実行など、すべてが自己責任となります。また、歴史的にセルフホスト版はクラウド版に比べて機能面で遅れをとる傾向があり（例：ダッシュボードの一部の機能が利用不可）、公式のサポートもSaaS版ほど手厚くはありません [45]。Supabase社のビジネスの焦点はあくまでSaaSプロダクトにあり、セルフホスティングのサポートはコミュニティ主導の側面が強いです [46]。

5.3 戦略的インプリケーション

Supabaseのセルフホスティングオプションは、単なるデプロイメントの選択肢以上の戦略的な意味を持ちます。それは、Firebaseの強力な武器である「ベンダーロックイン」に対する、Supabaseの**戦略的な「緊急脱出口（Escape Hatch）」**として機能しています。

Firebaseはプロプライエタリなプラットフォームであり、一度採用するとGoogleのインフラから離れることは極めて困難です [3]。この点が、多くの企業にとって採用を躊躇させる要因となっています。Supabaseは、オープンソースであることとセルフホスティングが可能であることを前面に押し出すことで、この懸念を直接的に解消します [2]。

開発者や企業は、「もし将来的にSaaSの価格が許容できなくなったり、SaaSでは満たせない特殊な要件が発生したりしても、自分たちで運用する道が残されている」という安心感を持って、まずは手軽なSaaS版を導入することができます。つまり、セルフホスティングの存在そのものが、より収益性の高いSaaSプロダクトの採用障壁を下げ、顧客がプラットフォームにコミットしやすくするための強力な保証となっているのです。この戦略が、SupabaseがFirebaseと効果的に競争するための重要な要

素の一つです。

第6章 市場の状況と競合上の位置づけ（2025年第3四半期）

BaaS市場は急速に進化しており、その中でSupabaseは独自の地位を確立しつつあります。本章では、2025年第3四半期時点でのSupabaseの市場における認知度、開発者からの評価、そして主要な競合製品との比較を通じて、その戦略的な位置づけを分析します。

6.1 市場での評価と開発者の採用状況

- 採用指標: 2025年現在、Supabaseは爆発的な成長を遂げています。登録開発者数は170万人を超え、GitHubのスター数は81,000以上、作成されたデータベースは100万を超えるとの報告があります [9]。Stack Overflowの2025年開発者調査によると、プロの開発者の6%がSupabaseを使用しており、市場への浸透が着実に進んでいることを示しています [11]。特にスタートアップからの支持は厚く、最近のY Combinatorのバッチでは40%の企業がSupabaseを基盤として利用しているとされています [10]。
- 資金調達と評価額: Supabaseは潤沢な資金力を有しており、2025年初頭にはシリーズDラウンドで2億ドルを調達し、評価額は20億ドルに達しました [9]。この強力な財務基盤は、市場からの高い期待と、継続的な製品開発および競争力を維持するためのリソースを保証するものです。年間経常収益（ARR）は7,000万ドルを目指しており、ビジネスとしても急成長しています [10]。
- コミュニティの評価: Hacker NewsやRedditといった開発者向けプラットフォームでは、Supabaseに対する評価は概ね非常に肯定的です。特に、オープンソースである点、SQLの強力さ、そして明瞭な価格体系が称賛されています [23]。一方で、一部機能の未成熟さ（例：認証クライアントのバグ、セルフホスト用ダッシュボードの機能不足）、RLSの学習曲線の険しさ、そして超大規模スケールでのコストに対する懸念といった批判的な意見も見られます [26]。
- 戦略的ポジショニング: Supabaseは、「vibe coding」という現代的な開発スタイルを標榜し、AIアプリケーション開発者など、スピードと直感的な体験を重視しつつも、本格的なデータベースのパワーを犠牲にしたくない層に強くアピールすることに成功しています [9]。

6.2 主要な代替製品の分析

Supabaseは多くの競合製品が存在する市場で戦っています。以下に主要な代替製品とその特徴を

挙げます。

- **Firebase**: 市場のリーダーであり、Supabaseが比較される際の基準となる存在です。Google Cloudとの緊密な統合、成熟したモバイル向け機能（プッシュ通知、ML Kitなど）、そして巨大なユーザーベースが強みです [59]。
- **Appwrite**: オープンソースのBaaSとしてSupabaseの直接的な競合です。認証、データベース、ストレージ、関数といった機能をバンドルで提供し、セルフホスティングも可能です。開発者体験の良さで評価されていますが、PostgreSQLベースではなく、Supabaseほどの成熟度はないと見なされることがあります [54]。
- **AWS Amplify**: AmazonのBaaS製品。AWSの広範なエコシステム（Cognito, DynamoDB, Lambdaなど）との深い統合が最大の強みです。非常にスケーラブルですが、SupabaseやFirebaseに比べて学習コストが高く、設定が複雑であると認識されています [54]。
- **Neon / PlanetScale**: これらはオールインワンのBaaSではなく、特化型のサーバーレスデータベースプロバイダーです（NeonはPostgreSQL、PlanetScaleはMySQL/Vitessベース）。データベース分岐（Branching）のような先進的な機能を提供し、Supabaseのデータベース部分と競合します [63]。これらのデータベースと、ClerkやAuth0のような認証サービスを組み合わせる「アンバンドル」なスタック構成がトレンドとなっています。
- **Hasura / Nhost**: GraphQLファーストのアプローチを取るプラットフォームです。Hasuraは既存のデータベース（PostgreSQLを含む）の上に瞬時にGraphQL APIを生成し、NhostはHasuraに認証やストレージをバンドルしてSupabaseに近い体験を提供します [3]。GraphQL中心のアーキテクチャを志向するチームにとって最適な選択肢です。

6.3 競合環境の比較表

プラットフォーム	基本理念	DBエンジン	強み	弱み	最適なユースケース	オープンソース
Supabase	統合型BaaS (PostgreSQL中心)	PostgreSQL	SQLのパワー、RLS、オープンソース、コスト予測性	モバイルネイティブ機能の不足、一部機能の未成熟さ	複雑なデータモデルを持つWebアプリ、スタートアップ	Yes
Firebase	統合型BaaS	NoSQL (Firestore	リアルタイム機能、	ベンダーロックイ	モバイルファースト	No

	(Google エコシステム)	)	モバイル連携、導入の容易さ	ン、NoSQLのクエリ制約、コストの予測不能性	のリアルタイムアプリ、MVP開発	
Appwrite	統合型 BaaS (自己完結型)	NoSQL (MariaDB)	シンプルな開発者体験、セルフホストの容易さ	PostgreSQL非採用、Supabaseよりエコシステムが小さい	フロントエンド開発者主導のプロジェクト、シンプルなバックエンド	Yes
AWS Amplify	統合型 BaaS (AWSエコシステム)	NoSQL (DynamoDB) 他	AWSサービスとの深い統合、エンタープライズ級のスケーラビリティ	学習コストが高い、設定が複雑	既存のAWSインフラを最大限活用したい大規模プロジェクト	Yes (一部)
Neon + Clerk	アンバンドル型 (Best-of-Breed)	Serverless PostgreSQL	DB分岐、各分野で最高の機能、柔軟なスタック構成	サービス間の連携を自己管理する必要がある、統合の複雑性	CI/CDワークフロー、最新技術スタックを求めるチーム	Yes

この市場分析から、BaaS市場が二極化している様子がうかがえます。一つは、Supabaseのようにデータベース、認証、ストレージといったバックエンド機能を**「再バンドル」し、統合されたプラットフォームとして提供する流れです。もう一つは、**Neon**（データベース）、**Clerk**（認証）、**Vercel**（ホスティング）のように、各分野で最高の機能を持つ特化型サービスを開発者が自由に組み合わせて利用する「アンバンドル」**の流れです。

Supabaseの長期的な成功は、この「再バンドル」された統合プラットフォームが、開発者自身が組み立てたカスタムスタックよりも優れた価値（生産性、保守性、コスト）を提供できるかどうかにかかっています。Supabaseの最大の競合は、Firebaseのような別の統合型プラットフォームだけでなく、専門

性の高いツール群の組み合わせそのものであると言えるでしょう。

第7章 Firebaseからの移行に関する戦略的考察

既存のFirebaseプロジェクトをSupabaseへ移行することは、多くの開発チームが検討する重要な戦略的選択肢です。しかし、この移行は単なる技術的な置き換えではなく、バックエンドアーキテクチャの根本的な再構築を意味します。本章では、移行によって得られる具体的なメリットと、それに伴う重大な課題やデメリットを詳細に分析し、移行を判断するためのフレームワークを提示します。

7.1 移行によって得られる定量的なメリット

FirebaseからSupabaseへの移行を検討する主な動機は、技術的および経済的な利点に集約されます。

- **SQLとクエリ能力の解放:** 移行の最大の技術的動機は、Firestoreのクエリ能力の制約からの脱却です。Supabaseに移行することで、複雑なJOIN、集計、トランザクションといったリレーショナルデータベースの全機能を活用できるようになります [12]。これにより、これまでクライアントサイドやCloud Functionsで複雑に実装していたデータ処理ロジックを、効率的な単一のSQLクエリに置き換えることができ、バックエンドのコードが簡素化され、パフォーマンスが向上する可能性があります。
- **コストの予測可能性と潜在的な削減:** Firebaseの操作ごとの従量課金モデルは、特に読み取り処理が多いアプリケーションや、利用量のスパイクが予測しにくい場合に、予期せぬ高額請求につながるリスクがあります [16]。Supabaseのリソースベースの価格体系は、月々のコストを予測しやすく、多くの場合、より経済的です [12]。ただし、一部のユーザーからは、超大規模スケールではSupabaseも高額になり得るとの指摘もあり、移行前には詳細なコストシミュレーションが不可欠です [22]。
- **ベンダーロックインの回避:** オープンソースのPostgreSQLを基盤とするSupabaseへの移行は、長期的な技術的自由を確保するための戦略的な一手です [3]。データは標準的な形式で容易にエクスポートでき、将来的にはオンプレミスや他のクラウドプロバイダー上でのセルフホスティングも可能です [2]。これにより、特定のベンダーの技術や価格戦略に縛られるリスクから解放されます。

7.2 移行における重大な課題とデメリット

メリットが大きい一方で、移行プロセスには多大な困難が伴います。これらを過小評価することは、プロジェクトの失敗に直結します。

- データモデルの全面的な再設計: これが移行における最大の障壁です。Firestoreの非正規化された階層的なNoSQL構造から、Supabaseの正規化されたリレーショナルなSQL構造へデータスキーマを変換する必要があります [1]。これは単なるデータのエクスポート/インポートではなく、データベース設計そのものをゼロからやり直す、大規模なアーキテクチャ作業です [14]。
- コアサービスの移行に伴う技術的困難:
  - データ移行: SupabaseはFirestoreからデータを移行するための公式スクリプト（firebase-to-supabaseリポジトリ）を提供しています [1]。しかし、これらはあくまで基本的なツールであり、ネストされたサブコレクションの扱いやデータ型の変換など、複雑な構造を持つデータには対応しきれない場合があります。GitHubのIssueトラッカーには、これらのツールに関する多くのバグ報告や機能改善要望が寄せられています [69]。
  - 認証情報の移行: ユーザーデータの移行は特に複雑です。パスワードハッシュのアルゴリズムの違い、Firebaseの文字列UIDとSupabaseのUUID形式の違い、複数の認証プロバイダーを持つユーザーの扱いなど、多くの課題が存在します [14]。一般的な戦略として、ユーザーが次回ログインするタイミングでデータを移行する「遅延移行（Lazy Migration）」が採用されることもありますが、これも実装が複雑です [71]。
  - ストレージの移行: Firebase Cloud Storageに保存されているすべてのファイルをダウンロードし、Supabase Storageに再アップロードする必要があります [12]。データ量が大きい場合、このプロセスには多大な時間とネットワーク転送コストがかかります。
- アプリケーションコードの全面的なリファクタリング: アプリケーションのデータアクセス層をすべて書き直す必要があります。Firebase SDKへのすべての呼び出し（CRUD操作、リアルタイムリスナー、認証ロジックなど）を、Supabaseクライアントライブラリの呼び出しに置き換える必要があります [12]。これはアプリケーションの規模によっては数ヶ月単位の作業となり得ます。
- 機能的なギャップ: Firebaseは、Supabaseよりも広範で成熟したエコシステムを持っています。移行すると、Firebase Cloud Messaging（プッシュ通知）、Remote Config、Crashlytics、Google Analyticsとのシームレスな連携といったネイティブ機能が利用できなくなります [13]。これらの機能は、サードパーティのサービスを新たに導入して代替する必要があります。

7.3 移行のフレームワークと最終提言

以上の分析から、FirebaseからSupabaseへの移行は、単なる「移植（Porting）」ではなく、**「再構築（Rebuild）」**であると捉えるべきです。この認識の転換は、移行プロジェクトの時間、コスト、リスクを正確に見積もる上で極めて重要です。

移行を成功させるための一般的なアプローチは、ダウンタイムを最小限に抑えるための**「並行稼働（Parallel Run）」**戦略です [12]。これは、一時的にFirebaseとSupabaseの両方のバックエンドを稼

働させ、まず書き込み処理を両方に送る「デュアルライト」を実装し、その後、読み取り処理を段階的にFirebaseからSupabaseに切り替えていくというものです。この戦略は安全ですが、根本的に異なる2つのデータベース間でデータの一貫性を維持する必要があるため、技術的な複雑性が非常に高くなります [14]。

最終的に、移行を決定するか否かは、以下のマトリクスを用いて、プロジェクト固有の状況に照らし合わせて判断することを推奨します。

考慮事項	Firebaseでの継続を推奨するシナリオ	Supabaseへの移行を推奨するシナリオ	判断のための主要な質問
クエリの複雑性	アプリケーションのクエリが単純なCRUD操作に留まる。	複雑なJOINや集計が頻繁に必要で、クライアント側のロジックが肥大化している。	現在のデータ取得ロジックは、将来の機能追加のボトルネックになっているか？
スケーラビリティ	予測可能なトラフィックで、現在のパフォーマンスに問題がない。	読み取りヘビーなワークロードで、Firestoreの読み取りコストが事業の収益性を圧迫している。	コストモデルはビジネスの成長と連動しているか？予測不能なコストスパイクのリスクは許容できるか？
開発予算と時間	移行のための大規模な再設計・開発リソースを確保できない。	長期的な技術的負債の解消と保守性向上のために、先行投資を行う戦略的判断ができる。	移行プロジェクトのROI（投資対効果）は、短期的な開発コストを正当化できるか？
チームの**SQL**習熟度	チームがNoSQLに習熟しており、SQLの経験が乏しい。	チームがPostgreSQLとリレーショナルデータベース設計に精通している。	チームはRLSのような新しいセキュリティパラダイムを学習し、正しく実装するスキルを持っているか？
ベンダーロックイン	Googleエコシステム内での完結にメリットを感じている。	オープンソース技術の採用とデータのポータビリティが、企業の長期的な技術	5年後、現在のプラットフォームから移行する必要が生じた場合、そのコストはど

		戦略に合致する。	の程度か？
モバイル機能要件	プッシュ通知やRemote Configなど、Firebaseのネイティブモバイル機能に強く依存している。	主要な機能がWeb中心であるか、プッシュ通知などをサードパーティサービスで代替する用意がある。	Firebase固有の機能は、アプリケーションのコアバリューにとってどの程度重要か？

結論として、FirebaseからSupabaseへの移行は、技術的な負債を解消し、長期的な柔軟性とコスト管理能力を獲得するための強力な選択肢です。しかし、それはバックエンドの全面的な再構築を伴う困難な道のりでもあります。移行の決定は、目先の課題解決だけでなく、アプリケーションの将来的な成長、チームのスキルセット、そして企業の技術戦略全体を俯瞰した上で、慎重に行われるべきです。

引用文献

1. Supabase vs Firebase, 9月 21, 2025にアクセス、https://supabase.com/alternatives/supabase-vs-firebase
2. Supabase vs. Firebase: a Complete Comparison in 2025 - Bytebase, 9月 21, 2025にアクセス、https://www.bytebase.com/blog/supabase-vs-firebase/
3. Firebase Alternatives to Consider in 2025 - DEV Community, 9月 21, 2025にアクセス、https://dev.to/riteshkokam/firebase-alternatives-to-consider-in-2025-456g
4. Find an Integration - Supabase, 9月 21, 2025にアクセス、https://supabase.com/partners/integrations
5. Artie | Works With Supabase, 9月 21, 2025にアクセス、https://supabase.com/partners/artie
6. BigQuery Wrapper | Works With Supabase, 9月 21, 2025にアクセス、https://supabase.com/partners/supabase_wrapper_bigquery
7. Self-Hosting with Docker | Supabase Docs, 9月 21, 2025にアクセス、https://supabase.com/docs/guides/self-hosting/docker
8. Self-Hosting | Supabase Docs, 9月 21, 2025にアクセス、https://supabase.com/docs/guides/self-hosting
9. Why Supabase Became the Go-To Open-Source Alternative to Firebase | Medium, 9月 21, 2025にアクセス、https://medium.com/@takafumi.endo/why-supabase-became-the-go-to-open-source-alternative-to-firebase-2d3cd59e7094
10. Supabase valuation, funding & news | Sacra, 9月 21, 2025にアクセス、https://sacra.com/c/supabase/
11. Technology | 2025 Stack Overflow Developer Survey, 9月 21, 2025にアクセス、https://survey.stackoverflow.co/2025/technology
12. How to Migrate from Firebase to Supabase (Step-by-Step Guide) - Supadex, 9月 21, 2025にアクセス、

https://www.supadex.app/blog/how-to-migrate-from-firebase-to-supabase-(step-by-step-guide)
13. Migrating from Firebase to Supabase (with FlutterFlow) – What should I know?, 9月 21, 2025にアクセス、https://community.flutterflow.io/ask-the-community/post/migrating-from-firebase-to-supabase-with-flutterflow---what-should-i-rI3teRVC70T0eJZ
14. How to migrate from Firebase to Supabase? - Emergence Engineering, 9月 21, 2025にアクセス、https://emergence-engineering.com/blog/firestore-supabase-migration
15. Supabase | The Postgres Development Platform., 9月 21, 2025にアクセス、https://supabase.com/
16. Supabase vs Firebase: Detailed Comparison for Your Next Project | by Harisudhan.S, 9月 21, 2025にアクセス、https://medium.com/@speaktoharisudhan/supabase-vs-firebase-detailed-comparison-for-your-next-project-ffcaaea30037
17. Supabase vs. Firebase: Which is best? [2025] - Zapier, 9月 21, 2025にアクセス、https://zapier.com/blog/supabase-vs-firebase/
18. Supabase vs. Firebase: Which BaaS is Best for Your App? - Netguru, 9月 21, 2025にアクセス、https://www.netguru.com/blog/supabase-vs-firebase
19. Firebase vs Supabase in 2025: Which one actually scales with you? - DEV Community, 9月 21, 2025にアクセス、https://dev.to/dev_tips/firebase-vs-supabase-in-2025-which-one-actually-scales-with-you-2374
20. Supabase vs Firebase: Which is Better? A Detailed Comparison - CloseFuture, 9月 21, 2025にアクセス、https://www.closefuture.io/blogs/supabase-vs-firebase
21. Supabase vs Firebase: Which BaaS Platform Leads in 2025? - Talent500, 9月 21, 2025にアクセス、https://talent500.com/blog/supabase-vs-firebase-baas-platform-comparison-2025/
22. Firebase vs Supabase: What are your NEGATIVE experiences or frustrations only? - Reddit, 9月 21, 2025にアクセス、https://www.reddit.com/r/FlutterDev/comments/1mtnr84/firebase_vs_supabase_what_are_your_negative/
23. Supabase has been better in every way than Firebase. Even if it wasn't open sour... | Hacker News, 9月 21, 2025にアクセス、https://news.ycombinator.com/item?id=26635781
24. Supabase vs. Firebase: Which solution Is better for authentication ? - Stack Overflow, 9月 21, 2025にアクセス、https://stackoverflow.com/beta/discussions/78581652/supabase-vs-firebase-which-solution-is-better-for-authentication
25. Supabase vs Firebase in 2025: The Ultimate BaaS Comparison - AnotherWrapper, 9月 21, 2025にアクセス、https://anotherwrapper.com/blog/supabase-vs-firebase
26. is Supabase that bad? - Reddit, 9月 21, 2025にアクセス、https://www.reddit.com/r/Supabase/comments/1ktkeh2/is_supabase_that_bad/
27. Can I use Supabase for analytics? - Tinybird, 9月 21, 2025にアクセス、

https://www.tinybird.co/blog-posts/can-i-use-supabase-for-user-facing-analytics
28. Best Practices for Securing and Scaling Supabase for Production Data Workloads | by firman brilian | Medium, 9月 21, 2025にアクセス、https://medium.com/@firmanbrilian/best-practices-for-securing-and-scaling-supabase-for-production-data-workloads-4394aba9e868
29. Tables and Data | Supabase Docs, 9月 21, 2025にアクセス、https://supabase.com/docs/guides/database/tables
30. Analytics Buckets | Supabase Docs, 9月 21, 2025にアクセス、https://supabase.com/docs/guides/storage/analytics/introduction
31. How to Connect Supabase to Snowflake: 2 Proven Methods - Estuary, 9月 21, 2025にアクセス、https://estuary.dev/blog/supabase-to-snowflake/
32. Supabase to BigQuery: Real-Time Sync in Minutes (No Code) | Estuary, 9月 21, 2025にアクセス、https://estuary.dev/blog/supabase-to-bigquery/
33. Google BigQuery and Supabase: Automate Workflows with n8n, 9月 21, 2025にアクセス、https://n8n.io/integrations/google-bigquery/and/supabase/
34. Integrate Supabase with Google Bigquery for automation | viaSocket, 9月 21, 2025にアクセス、https://viasocket.com/integrations/supabase/google-bigquery
35. Supabase Analytics Buckets with Iceberg Support, 9月 21, 2025にアクセス、https://supabase.com/blog/analytics-buckets
36. PostgreSQL vs TiDB: Key Differences and Use Cases - w3resource, 9月 21, 2025にアクセス、https://www.w3resource.com/PostgreSQL/snippets/postgresql-vs-tidb.php
37. TiDB vs. CockroachDB: the Distributed Clash between MySQL and PostgreSQL - Bytebase, 9月 21, 2025にアクセス、https://www.bytebase.com/blog/tidb-vs-cockroachdb/
38. TiDB vs Traditional Databases: Scalability and Performance, 9月 21, 2025にアクセス、https://www.pingcap.com/article/tidb-vs-traditional-databases-scalability-and-performance/
39. Pricing & Fees | Supabase, 9月 21, 2025にアクセス、https://supabase.com/pricing
40. TiDB Cloud Reviews - AWS Marketplace, 9月 21, 2025にアクセス、https://aws.amazon.com/marketplace/reviews/reviews-list/prodview-7xendfnh6ykg2
41. TiDB Reviews 2025: Details, Pricing, & Features - G2, 9月 21, 2025にアクセス、https://www.g2.com/products/tidb/reviews
42. Cutting over: Our journey from AWS Aurora MySQL to TiDB - Plaid, 9月 21, 2025にアクセス、https://plaid.com/blog/switching-to-tidb/
43. Supabase on the AWS Marketplace, 9月 21, 2025にアクセス、https://supabase.com/blog/supabase-aws-marketplace
44. Supabase vs Firebase: Choosing the Right Backend for Your Next Project - Jake Prins, 9月 21, 2025にアクセス、https://www.jakeprins.com/blog/supabase-vs-firebase-2024
45. Your experience with Supabase - Reddit, 9月 21, 2025にアクセス、https://www.reddit.com/r/Supabase/comments/1fhnydy/your_experience_with_su

pabase/
46. The Case For Better Self-hosting Supabase Support | by Sachin Agarwal | Medium, 9月 21, 2025にアクセス、https://medium.com/@sachin_49501/the-case-for-better-self-hosting-supabase-support-dbdab63df3fa
47. Self Hosting Supabase on AWS - DEV Community, 9月 21, 2025にアクセス、https://dev.to/aws-builders/self-hosting-supabase-on-aws-1cdl
48. Supabase on AWS - Why would anyone want to host it? - Devopsity, 9月 21, 2025にアクセス、https://devopsity.io/blog/supabase-on-aws-why-would-anyone-want-to-host-it/
49. Supabase on Ubuntu 24.04 LTS | Support by SupportedImages - AWS Marketplace, 9月 21, 2025にアクセス、https://aws.amazon.com/marketplace/pp/prodview-bolyo34t4sdn2
50. Self-hosted Supabase on AWS - GitHub, 9月 21, 2025にアクセス、https://github.com/supabase-community/supabase-on-aws
51. Supabase: Open Source Firebase Alternative with Hosted Postgres Backend - AWS Marketplace, 9月 21, 2025にアクセス、https://aws.amazon.com/marketplace/pp/prodview-qg3lale77sic4
52. The Case for better Self-Hosted Supabase Support - Reddit, 9月 21, 2025にアクセス、https://www.reddit.com/r/Supabase/comments/10v7sk7/the_case_for_better_self_hosted_supabase_support/
53. Supabase seems too good to be true, someone steelman other options? - Reddit, 9月 21, 2025にアクセス、https://www.reddit.com/r/Supabase/comments/18lvcrf/supabase_seems_too_good_to_be_true_someone/
54. 2025's Best Firebase Alternatives: Open-Source & Scalable Solutions, 9月 21, 2025にアクセス、https://blog.back4app.com/firebase-alternatives/
55. Supabase raises $200M Series D at $2B valuation | Hacker News, 9月 21, 2025にアクセス、https://news.ycombinator.com/item?id=43763225
56. We currently make use of Supabase and it's been fantastic. It's enabled us to co... | Hacker News, 9月 21, 2025にアクセス、https://news.ycombinator.com/item?id=29405501
57. Supabase is amazing - Reddit, 9月 21, 2025にアクセス、https://www.reddit.com/r/Supabase/comments/1ej5dck/supabase_is_amazing/
58. This guy had a problem with Supabase when it wasn't GA yet and now says he'll never use it again. How is it Supabase's fault? - Reddit, 9月 21, 2025にアクセス、https://www.reddit.com/r/Supabase/comments/1g5nzkp/this_guy_had_a_problem_with_supabase_when_it/
59. 10+ Best Open Source Supabase Alternatives (2025) - OpenAlternative, 9月 21, 2025にアクセス、https://openalternative.co/alternatives/supabase
60. Supabase and Its 10 Alternatives - Infyways Solutions, 9月 21, 2025にアクセス、https://www.infyways.com/supabase-alternatives/
61. Best Backend as a Service (BaaS) Platforms - Galaxy, 9月 21, 2025にアクセス、https://www.getgalaxy.io/learn/data-tools/best-backend-as-a-service-baas-platf

orms
62. Top 7 Firebase Alternatives for App Development in 2025 - SigNoz, 9月 21, 2025にアクセス、https://signoz.io/comparisons/firebase-alternatives/
63. Supabase Alternatives in 2025 - DEV Community, 9月 21, 2025にアクセス、https://dev.to/bytebase/supabase-alternatives-in-2025-1p8g
64. 10 Supabase Alternatives for Backend Developers in 2025 - Galaxy, 9月 21, 2025にアクセス、https://www.getgalaxy.io/resources/supabase-alternatives-2025
65. Top 10 Supabase alternatives to fuel your app-building process in 2025 - ToolJet Blog, 9月 21, 2025にアクセス、https://blog.tooljet.ai/supabase-alternatives/
66. Migrating from Firebase Firestore to Supabase: A Complete Guide | by Abhi Patel | Medium, 9月 21, 2025にアクセス、https://medium.com/@apate337/migrating-from-firebase-firestore-to-supabase-a-complete-guide-ed21fd4e22d6
67. Migrate from Firebase Firestore to Supabase, 9月 21, 2025にアクセス、https://supabase.com/docs/guides/platform/migrating-to-supabase/firestore-data
68. Firebase to Supabase Migration Guide - GitHub, 9月 21, 2025にアクセス、https://github.com/supabase-community/firebase-to-supabase
69. Issues · supabase-community/firebase-to-supabase - GitHub, 9月 21, 2025にアクセス、https://github.com/supabase-community/firebase-to-supabase/issues
70. How I Migrated 182000 Users to Supabase | by Tarik - Level Up Coding, 9月 21, 2025にアクセス、https://levelup.gitconnected.com/how-i-migrated-182k-users-to-supabase-e3c1fb0ed7bf
71. Migrate from Firebase Auth to Supabase, 9月 21, 2025にアクセス、https://supabase.com/docs/guides/platform/migrating-to-supabase/firebase-auth
72. How I Migrated 1000+ Users from Firebase to Supabase | Fleeting Notes, 9月 21, 2025にアクセス、https://www.fleetingnotes.app/posts/migrating-from-firebase-to-supabase
73. Migrated from Firebase Storage to Supabase, 9月 21, 2025にアクセス、https://supabase.com/docs/guides/platform/migrating-to-supabase/firebase-storage