Ordivis

TEIL VI · BETRIEB

Datenbank-Leistung, Verbindungspools & Dimensionierung

Datenbank-Leistung, Verbindungspools und Dimensionierung je Kundengröße

Produktversion 2026.09.20 · Handbuch-Revision 1.0
Stand 20.09.2026 · Plattform .NET 10 / Windows Server

ID-51 Leistung

Datenbank-Leistung, Verbindungspools & Dimensionierung

Dieses Kapitel richtet sich an **IT-Betrieb / Datenbank-Administratoren**. Es beschreibt jede leistungsrelevante Einstellung im Detail und gibt **empfohlene Werte je Kundengröße** (nach Lizenzstufe) an. Es baut auf den Systemvoraussetzungen (ID-02) und dem Betriebshandbuch (ID-50, Abschnitt 2.3) auf.

INHALT

1 Grundprinzip: zwei Ebenen, ein Zusammenspiel

Ordivis IT betreibt jeden Mandanten in einer **eigenen Datenbank** (`infradesk_<slug>`) auf **einem** PostgreSQL-Server (Database-per-Tenant), plus eine zentrale, mandantenfreie **Registry-Datenbank** (`infradesk`). API und Self-Service-Portal laufen in der Standardinstallation auf **derselben Maschine** wie PostgreSQL.

Für die Leistung zählen zwei Ebenen, die zusammenpassen müssen:

- **Server (PostgreSQL):** `max_connections` ist die **harte Obergrenze** aller gleichzeitigen Verbindungen. Jede Verbindung ist bei PostgreSQL ein **eigener Prozess** und kostet Arbeitsspeicher.
- **Client (Npgsql, im API-Dienst):** Npgsql hält je Datenbank einen **eigenen Verbindungspool**. Der Pool bestimmt, **wie schnell** die harte Obergrenze erreicht wird.

> **Wichtig:** Die Zahl der **Mitarbeiter** bestimmt die Verbindungslast **nicht** direkt. Alle Anwender (Agenten im Client, Melder im Portal) laufen über die **API** – diese bündelt sie über ihren Pool. Eine Verbindung wird je Anfrage nur für **Millisekunden** gehalten und danach zurückgegeben. Der eigentliche Engpass skaliert mit der Zahl der **gleichzeitig aktiven Mandanten**, nicht mit der Kopfzahl. Ein Einzelkunde mit 1.000 Mitarbeitern ist **ein** Mandant und unkritisch; kritisch wird es erst bei **vielen** Mandanten (MSP/Kommune).

2 PostgreSQL-Server-Parameter

2.1 Wie Ordivis IT sie setzt

Bringt der Installer PostgreSQL **selbst** mit, berechnet er die Werte aus der Zielmaschine (Arbeitsspeicher, Prozessorkerne, freier Plattenplatz) und setzt sie per `ALTER SYSTEM` – geschrieben nach `postgresql.auto.conf`. Ihre handgepflegte `postgresql.conf` bleibt unberührt, und jeder Wert lässt sich mit `ALTER SYSTEM RESET <name>` einzeln zurücknehmen (Details in ID-50, Abschnitt 2.3). Eine **mitgebrachte/externe** Instanz wird **nicht** angefasst – dort setzen Sie die Werte selbst (Abschnitt 6).

2.2 Die Parameter im Einzelnen

Parameter	Bedeutung	Herleitung (Installer)	Wirksam ab
`max_connections`	Höchstzahl gleichzeitiger Verbindungen. Jede = ein Prozess.	`8 × GB RAM`, begrenzt auf **100–400**	**Neustart**
`shared_buffers`	Von PostgreSQL selbst verwalteter Cache.	`25 % RAM`, gedeckelt auf **8 GB**	**Neustart**
`effective_cache_size`	*Planer-Hinweis*, wie viel Cache (inkl. OS) verfügbar ist – belegt keinen Speicher.	**50 % RAM** (nicht 75 %, da API und Portal mitlaufen)	Reload
`work_mem`	Speicher **je Sortier-/Hash-Schritt je Verbindung**. Der klassische Weg in den Speicherüberlauf.	`(25 % RAM) ÷ (max_connections × 2)`, **4–16 MB**	Reload
`maintenance_work_mem`	Speicher für Wartung (Index-Aufbau, `VACUUM`). Läuft selten parallel.	`10 % RAM`, gedeckelt auf **2 GB**	Reload
`max_wal_size` / `min_wal_size`	Größe des Write-Ahead-Logs. Größer = weniger Checkpoints, mehr Plattenbedarf.	nach freiem Platz: 1 / 4 / 8 / 16 GB	Reload
`idle_in_transaction_session_timeout`	Trennt Verbindungen, die eine Transaktion offen halten, ohne zu arbeiten.	**60 s**	Reload
`shared_preload_libraries`	Lädt `pg_stat_statements` (Grundlage der Abfrage-Analyse).	ergänzt (überschreibt nicht)	**Neustart**

> **Achtung:** `max_connections` und `shared_buffers` werden erst nach einem **vollständigen Neustart** des Dienstes wirksam – ein `reload` genügt **nicht**. Die übrigen Parameter greifen per `reload` (`SELECT pg_reload_conf();`).

2.3 Bewusst NICHT gesetzt: `statement_timeout`

Ordivis IT setzt **kein** globales `statement_timeout`. Grund: Die Datenbank-**Migrationen** laufen unter derselben Rolle wie die Anwendung (`infradesk_app`). Ein globales oder rollenweites Zeitlimit würde damit nicht nur lange Abfragen, sondern auch **Index-Neuaufbauten beim Update**, `pg_dump`-Sicherungen und die Größenabfrage des Monitorings abbrechen. Ein Zeitlimit gehört dorthin, wo der Aufrufer bekannt ist – als Npgsql-`CommandTimeout` je Abfrage.

2.4 Werte ändern und zurücknehmen

```
-- Wert setzen (Beispiel), schreibt nach postgresql.auto.conf
ALTER SYSTEM SET max_connections = '300';
-- Einzelnen Wert wieder auf den Standard zurücknehmen
ALTER SYSTEM RESET max_connections;
-- Reload-fähige Parameter ohne Neustart übernehmen
SELECT pg_reload_conf();
```

Für einen Neustart-pflichtigen Parameter anschließend den Dienst neu starten:

```
Restart-Service postgresql-ordivis
```

Das Tuning-Skript lässt sich jederzeit erneut anstoßen (idempotent), zum reinen Anzeigen mit `-DryRun`:

```
powershell -File tune-postgresql.ps1 -PgSuperPassword "<Kennwort>" -DryRun
```

3 Der Verbindungspool (`Database:MaxPoolSizePerTenant`)

3.1 Warum es diese Einstellung gibt

Npgsql führt **je Connection-String einen eigenen Pool**. Bei Database-per-Tenant bedeutet das: **je Mandant ein eigener Pool**. Ohne Grenze poolt Npgsql je Mandant mit dem Standardwert **100** – zwei aktive Mandanten schöpfen ein ungetuntes `max_connections = 100` also bereits rechnerisch aus. Deshalb begrenzt Ordivis IT den Pool **je Mandant**.

3.2 Die Einstellung im Detail

In `appsettings.json` des API-Dienstes:

```
"Database": {
  "MaxPoolSizePerTenant": 50
}
```

Aspekt	Verhalten
Bedeutung	Obergrenze (`Maximum Pool Size`) je Mandanten-Pool **und** für die zentrale Registry-/Haupt-DB.
Standardwert	**50** (gilt auch, wenn der Schlüssel fehlt).
`≤ 0`	Bewusst **unbegrenzt** – fällt auf den Npgsql-Standard 100 zurück.
Wo erzwungen	Zentral beim Auflösen der Mandanten-Verbindung. Greift dadurch **auch für bereits angelegte Mandanten** (deren gespeicherter String die Grenze noch nicht trägt), nicht nur für neue.
Untergrenze	Npgsqls `Minimum Pool Size` bleibt **0** – **leerlaufende** Mandanten-Pools geben ihre Verbindungen nach ~5 min wieder frei.

> **Gut zu wissen:** Weil leerlaufende Pools sich leeren (`Minimum Pool Size = 0`), zählt für die Dimensionierung nur die Zahl der **gleichzeitig aktiven** Mandanten – nicht die Zahl aller angelegten.

3.3 Ändern

`Database:MaxPoolSizePerTenant` in der `appsettings.json` des API-Dienstes anpassen und den Dienst neu starten (die Grenze wird beim Verbindungsaufbau angewendet):

```
Restart-Service InfraDesk.API
```

4 Dimensionierung: die Formel

Die Client-Pools müssen zur Server-Obergrenze passen. Faustformel für die **theoretische Spitze**:

> **Empfehlung – Dimensionierungsregel:** `max_connections ≥ (gleichzeitig aktive Mandanten + 1) × MaxPoolSizePerTenant + Reserve`
> Das `+ 1` steht für die zentrale Registry-/Haupt-DB. Die **Reserve** (~15–20) deckt die für Superuser reservierten Verbindungen (`superuser_reserved_connections` , Standard 3), Sicherungen (`pg_dump`), das Monitoring und die manuelle Administration ab.

Nach dem Pool aufgelöst:

`MaxPoolSizePerTenant ≤ (max_connections − Reserve) ÷ (gleichzeitig aktive Mandanten + 1)`

Daraus ergeben sich **zwei Betriebsformen**:

- **Wenige große Mandanten** (Corporate/Kommune-Einzelmandant): großzügiger Pool, da wenige Pools den Server teilen.
- **Viele kleine Mandanten** (MSP, Kommune mit vielen Untermandanten): **kleiner** Pool, da sich viele Pools die Obergrenze teilen. Alternativ die Mandanten auf **mehrere DB-Server** verteilen.

Ein vorgeschalteter Pooler (z. B. PgBouncer) löst das bei Database-per-Tenant **nicht**: Er poolt je (Benutzer, Datenbank) – aus einem Pool würden **N** Pools. Dieselbe Wirkung erzielt die eine Zeile `MaxPoolSizePerTenant` . (PgBouncer ist nur in der optionalen HA-Topologie mit **einer** Haupt-DB sinnvoll.)

5 Empfohlene Einstellungen je Kundengröße (Lizenz)

Die PostgreSQL-Spalten sind die Werte, die der Installer auf **selbst installierten** Servern automatisch berechnet – zugleich die **Zielwerte**, wenn Sie eine externe Instanz von Hand einstellen (Abschnitt 6). Die Hardware-Angaben knüpfen an die Basislinie aus ID-02 an (Minimum 8 GB / 4 Kerne).

5.1 Corporate (Privatwirtschaft)

Ein Unternehmen = **ein** Mandant (Ausnahme: Konzern mit bewusst getrennten Untermandanten).

Lizenzstufe (Admins/Agenten)	RAM	Kerne	`max_connections`	`shared_buffers`	`effective_cache_size`	`work_mem`	MaxPoolSizePerTenant
SMB (5 / 2)	8 GB	4	100	2 GB	4 GB	10 MB	**30**

Lizenzstufe (Admins/Agenten)	RAM	Kerne	`max_connections`	`shared_buffers`	`effective_cache_size`	`work_mem`	MaxPoolSizePerTenant
MID (20 / 5)	16 GB	4–8	128	4 GB	8 GB	16 MB	**50** (Standard)
Large (30 / 10)	32 GB	8	256	8 GB	16 GB	16 MB	**80**
Enterprise (∞ / ∞)	64 GB	16	400	8 GB[1]	32 GB	16 MB	**100** (1 Mandant)

[1] `shared_buffers` ist bei 8 GB gedeckelt. Auf einem **dedizierten** DB-Server (ohne API und Portal) darf er höher, und `effective_cache_size` auf ~75 % steigen.

> **Hinweis SMB:** Auf der 8-GB-Minimalausstattung ist `max_connections` nur 100. Der empfohlene Pool von **30** hält bewusst Reserve frei; bei der geringen Last eines SMB-Mandanten ist auch der Standard 50 unkritisch. **Enterprise mit mehreren Untermandanten:** den Pool nach der Formel aus Abschnitt 4 teilen (Richtwert **50** je Mandant bei bis zu ~6 gleichzeitig aktiven).

5.2 Kommunen (Gov)

Kommunen betreiben häufig **mehrere** Mandanten (Kernverwaltung, Eigenbetriebe, Verwaltungsgemeinschaft). Die Lizenz staffelt nach Einwohnern; für die Dimensionierung zählt die **Mandantenzahl**.

Größe (Einwohnerstufe)	RAM	Kerne	`max_connections`	gleichz. aktive Mandanten	MaxPoolSizePerTenant
Klein (Stufe 1–3)	8–16 GB	4	100–128	1–3	**25–30**
Mittel (Stufe 4–6)	16–32 GB	4–8	128–256	3–8	**20–25**
Groß (Stufe 7–9)	32–64 GB	8–16	256–400	5–15	**15–25** (nach Formel)

> **Empfehlung:** Den Pool aus `max_connections` und der **erwarteten Zahl gleichzeitig aktiver Mandanten** herleiten (Abschnitt 4), nicht raten. Beispiel Stufe 6: 32 GB → `max_connections` 256, ~8 aktive Mandanten → `(256 − 20) ÷ (8 + 1) ≈ 26` → **25**.

5.3 MSP (viele Kundenmandanten)

Der Managed-Service-Provider betreibt **viele** Kundenmandanten auf gemeinsamer Infrastruktur. Hier ist ein **kleiner** Pool entscheidend.

Szenario	RAM	Kerne	`max_connections`	gleichz. aktive Mandanten	MaxPoolSizePerTenant
Einstieg	32 GB	8	256	~20 (von z. B. 100 angelegten)	**8–10**
Wachstum	64 GB	16	400	~40	**7–8**

> **Achtung:** Bei sehr vielen Mandanten reicht ein einzelner Server nicht beliebig weit. Faustregel: Übersteigt die Zahl **gleichzeitig aktiver** Mandanten die durch `max_connections` tragbare Menge (bei kleinstem sinnvollen Pool ~5–6), die Mandanten auf **mehrere DB-Server/Knoten** verteilen. Nur gleichzeitig aktive zählen – leerlaufende Pools geben ihre Verbindungen frei (Abschnitt 3.2).

6 Mitgebrachte / externe PostgreSQL-Instanz

> 🔑 Datenbank-Administrator auf der PostgreSQL-Instanz — nicht in Client oder Portal, sondern unmittelbar an der Datenbank (`psql` oder ein gleichwertiges Werkzeug).

Ist im Setup eine **externe** Datenbank gewählt oder läuft bereits ein PostgreSQL, tunt der Installer diese Instanz **nicht** – ihre Konfiguration gehört Ihnen.

🔒 **Pflichtreihenfolge** — `max_connections` und `shared_buffers` greifen **erst nach einem Neustart** des Dienstes. Wer ihn auslässt, misst danach die alten Werte und sucht am falschen Ende.

1. Ermitteln Sie die Größenstufe Ihrer Instanz nach Abschnitt 5.
2. Setzen Sie die Zielwerte, hier am Beispiel der Stufe *Large* (32 GB):

```
ALTER SYSTEM SET max_connections = '256';
ALTER SYSTEM SET shared_buffers = '8GB';
ALTER SYSTEM SET effective_cache_size = '16GB';
ALTER SYSTEM SET work_mem = '16MB';
ALTER SYSTEM SET maintenance_work_mem = '2GB';
ALTER SYSTEM SET idle_in_transaction_session_timeout = '60s';
-- danach: Dienst neu starten (max_connections/shared_buffers brauchen das)
```

`MaxPoolSizePerTenant` wird unabhängig davon **immer** in der `appsettings.json` gesetzt – die Client-Grenze ist keine Server-Einstellung.

> **Wichtig:** Achten Sie darauf, dass `max_connections` der **externen** Instanz zur Summe Ihrer Pools passt (Formel in Abschnitt 4). Teilt sich die Instanz mit **anderen** Anwendungen den Server, deren Verbindungen von Ihrem Budget abziehen.

7 Prüfen & Überwachen

Aktuelle Werte am Server auslesen:

```
SELECT name, setting, unit FROM pg_settings
WHERE name IN ('max_connections','shared_buffers','effective_cache_size','work_mem','max_wal_size');

-- Momentane Verbindungslast (gesamt gegen die Obergrenze)
SELECT count(*) AS aktiv,
       current_setting('max_connections')::int AS obergrenze
FROM pg_stat_activity;

-- Verbindungen je Datenbank (= je Mandant)
SELECT datname, count(*) FROM pg_stat_activity GROUP BY datname ORDER BY 2 DESC;
```

Im Produkt: Die SuperAdmin-**Plattformüberwachung** zeigt die Verbindungslast in Prozent der `max_connections`. Standard-Warnschwellen (über `Platform:Thresholds` anpassbar):

Schwelle	Warnung	Kritisch
Verbindungslast (% von `max_connections`)	**80 %**	**90 %**
Plattenfüllstand	85 %	92 %
Langlaufende Abfragen (> 30 s)	ab 3	–

Erreicht die Verbindungslast dauerhaft die Warnschwelle, ist entweder der Pool zu groß für die Mandantenzahl oder `max_connections` (bzw. die Hardware) zu klein – siehe Formel in Abschnitt 4.

8 Symptome & Diagnose

Symptom	Ursache	Abhilfe
`FATAL: sorry, too many clients already` / `remaining connection slots are reserved`	`max_connections` ausgeschöpft: Zahl der Mandanten × Pool zu hoch.	`MaxPoolSizePerTenant` senken **oder** `max_connections` /RAM erhöhen (Formel Abschnitt 4).
Client-Zeitüberschreitungen beim Verbindungsaufbau, obwohl der Server nicht ausgelastet ist	Pool eines einzelnen Mandanten zu **klein** für seine Lastspitze.	`MaxPoolSizePerTenant` erhöhen (sofern `max_connections` Reserve hat).
Plattform-Monitoring meldet Verbindungslast > **80 %**	Nähert sich der Obergrenze.	Frühwarnung – Dimensionierung prüfen, bevor es zu Ablehnungen kommt.
Server wird beim Start/Update langsam, viel Speicherverbrauch	`max_connections` × `work_mem` zu hoch für den RAM.	`work_mem` prüfen (wird je Sortier-/Hash-Schritt je Verbindung belegt), ggf. senken.
Nach Änderung von `max_connections` keine Wirkung	Nur `reload` statt Neustart ausgeführt.	Dienst `postgresql-ordivis` neu starten (Abschnitt 2.2).

Verwandte Themen

- **ID-50 – Betriebshandbuch – Installation & Serverbetrieb** – Installation und Betrieb der Datenbank

- **ID-02 – Systemvoraussetzungen & Architektur** – Hardware- und Netzvoraussetzungen
- **ID-52 – Datenbank-Referenz** – Tabellen, Indizes und Feldstruktur
- **ID-27a – Reports & SQL-Abfragen** – Abfragen, die Last erzeugen